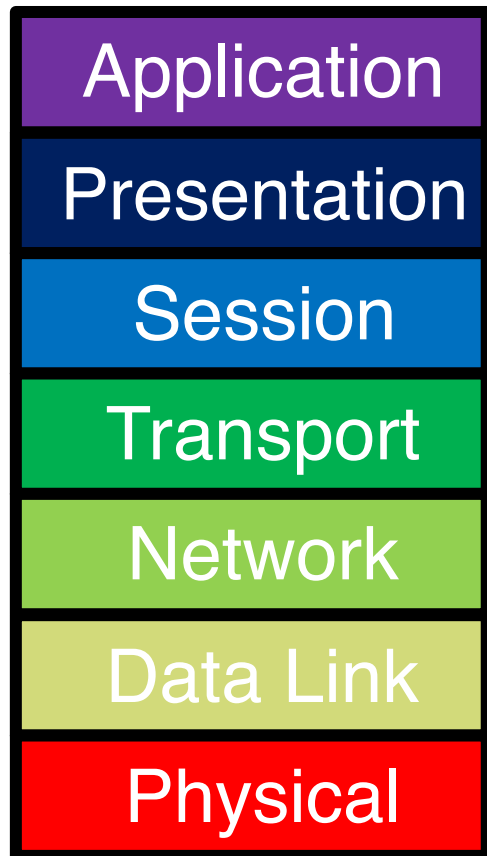


# CS4700/5700: Network fundamentals

Internetworking.

# Network Layer

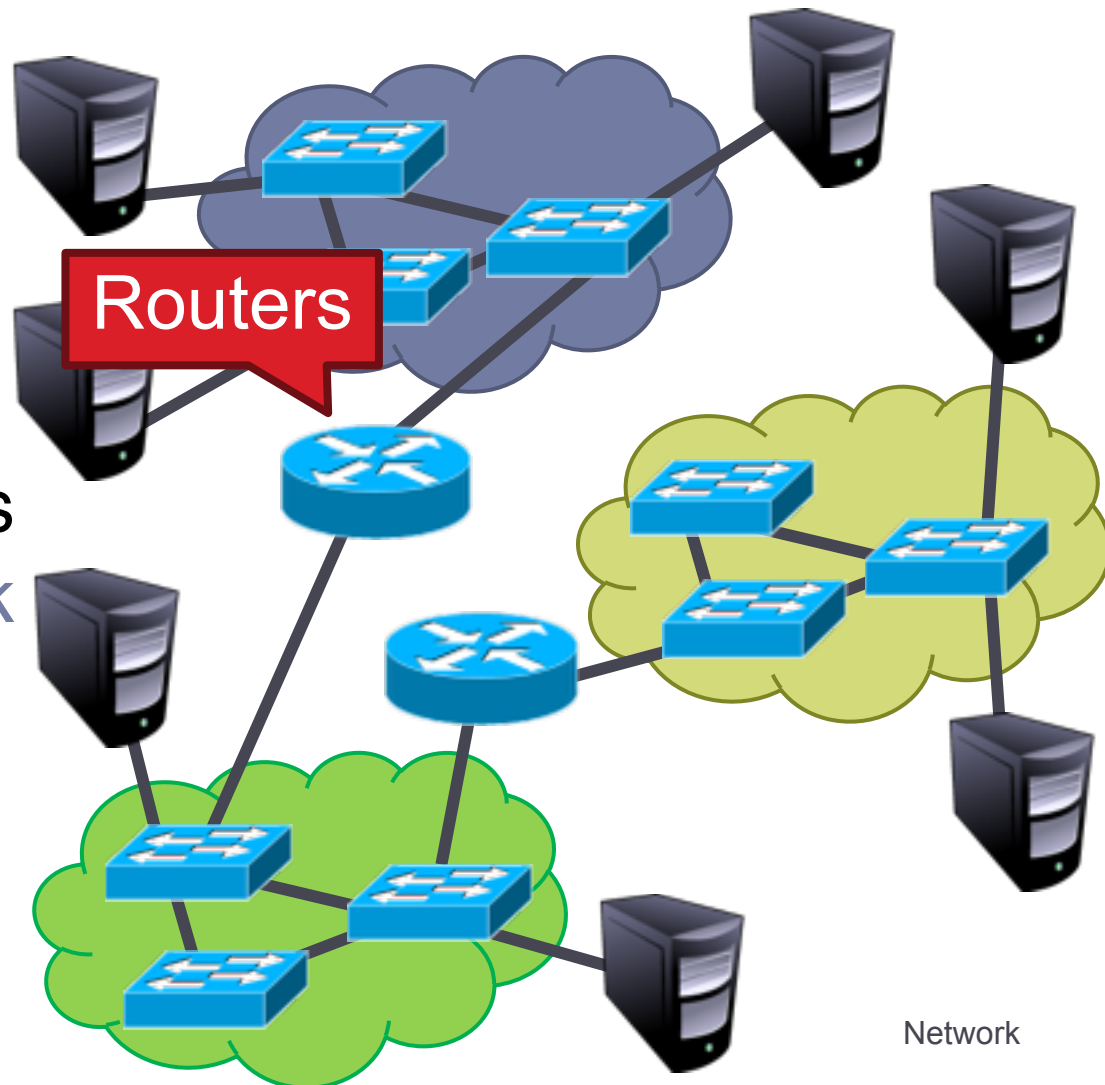
---



- ▶ **Function:**
  - ▶ Route packets end-to-end on a network, through multiple hops
- ▶ **Key challenge:**
  - ▶ How to represent addresses
  - ▶ How to route packets
    - ▶ Scalability
    - ▶ Convergence

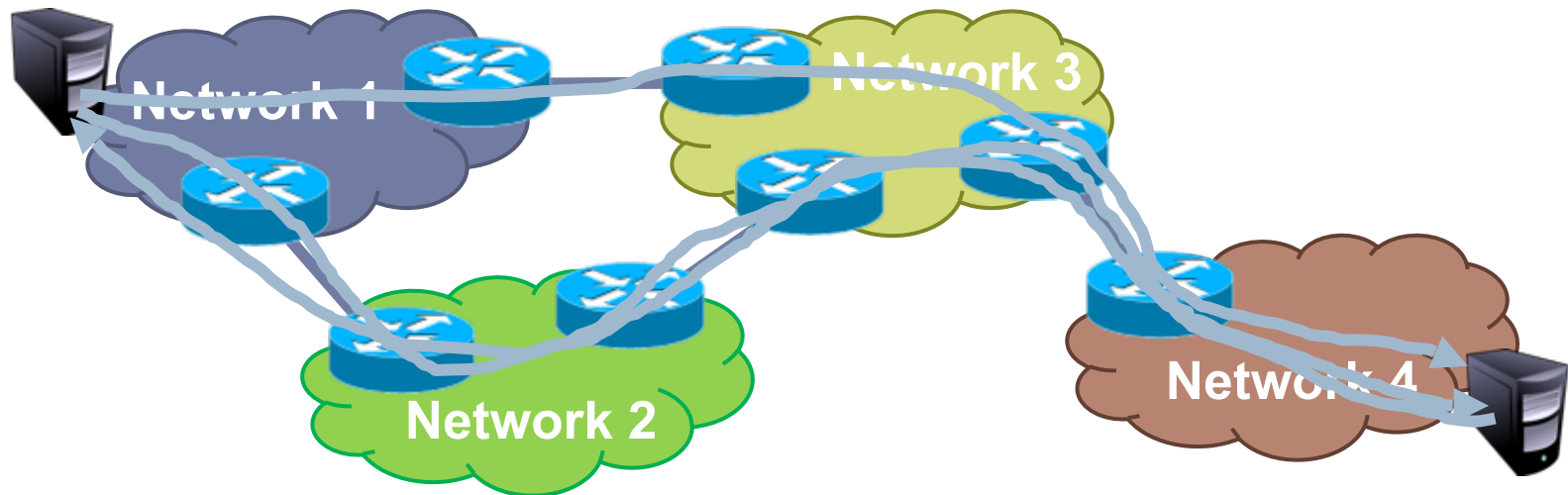
# Routers, Revisited

- ▶ How to connect multiple LANs?
- ▶ LANs may be incompatible
  - ▶ Ethernet, Wifi, etc...
- ▶ Connected networks form an **internetwork**
  - ▶ The Internet is the best known example



# Structure of the Internet

---



- ▶ Ad-hoc interconnection of networks
  - ▶ No organized topology
  - ▶ Vastly different technologies, link capacities
- ▶ Packets travel end-to-end by hopping through networks
  - ▶ Routers “peer” (connect) different networks
  - ▶ Different packets may take different routes

# Internetworking Issues

---

- ▶ **Naming / Addressing**
  - ▶ How do you designate hosts?
- ▶ **Routing**
  - ▶ Must be **scalable** (i.e. a switched Internet won't work)
- ▶ **Service Model**
  - ▶ What gets sent?
  - ▶ How fast will it go?
  - ▶ What happens if there are failures?
  - ▶ Must deal with **heterogeneity**
    - ▶ Remember, every network is different



# 1: Addressing

# Possible Addressing Schemes

---

## ▶ Flat

- ▶ e.g. each host is identified by a 48-bit MAC address
- ▶ Router needs an entry for every host in the world
  - ▶ Too big
  - ▶ Too hard to maintain (hosts come and go all the time)
  - ▶ Too slow (more later)

# Possible Addressing Schemes

---

## ▶ Flat

- ▶ e.g. each host is identified by a 48-bit MAC address
- ▶ Router needs an entry for every host in the world
  - ▶ Too big
  - ▶ Too hard to maintain (hosts come and go all the time)
  - ▶ Too slow (more later)

## ▶ Hierarchy

- ▶ Addresses broken down into segments
- ▶ Each segment has a different level of specificity



# Example: Telephone Numbers

---

1-617-373-1234

# Example: Telephone Numbers

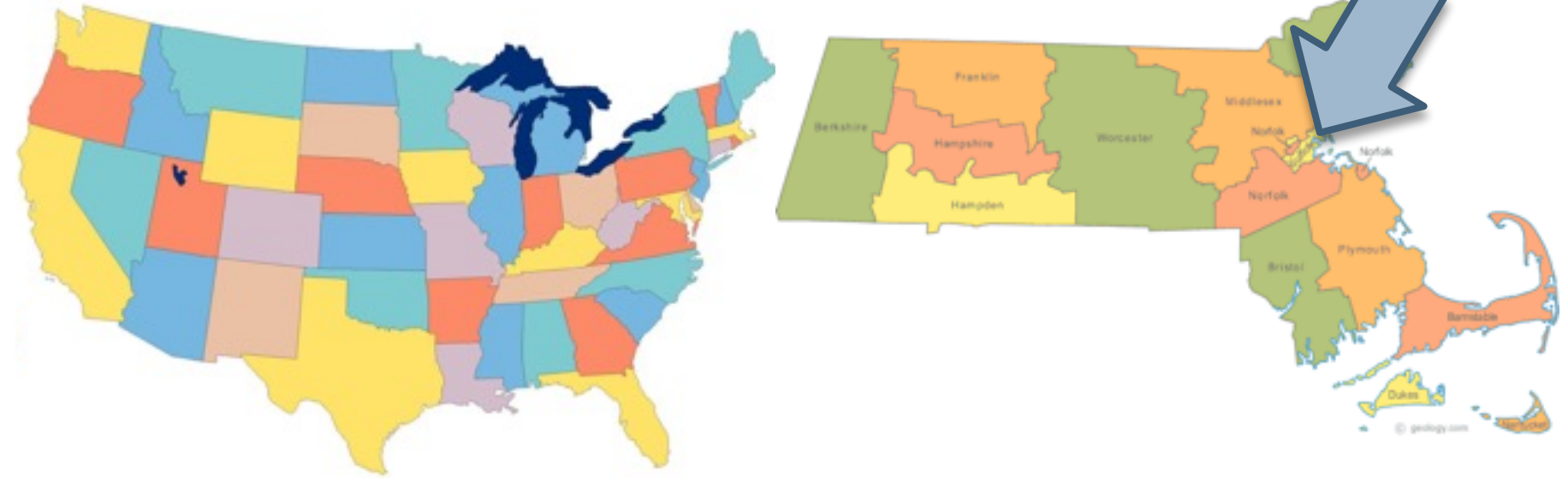
---

1-617-373-1234



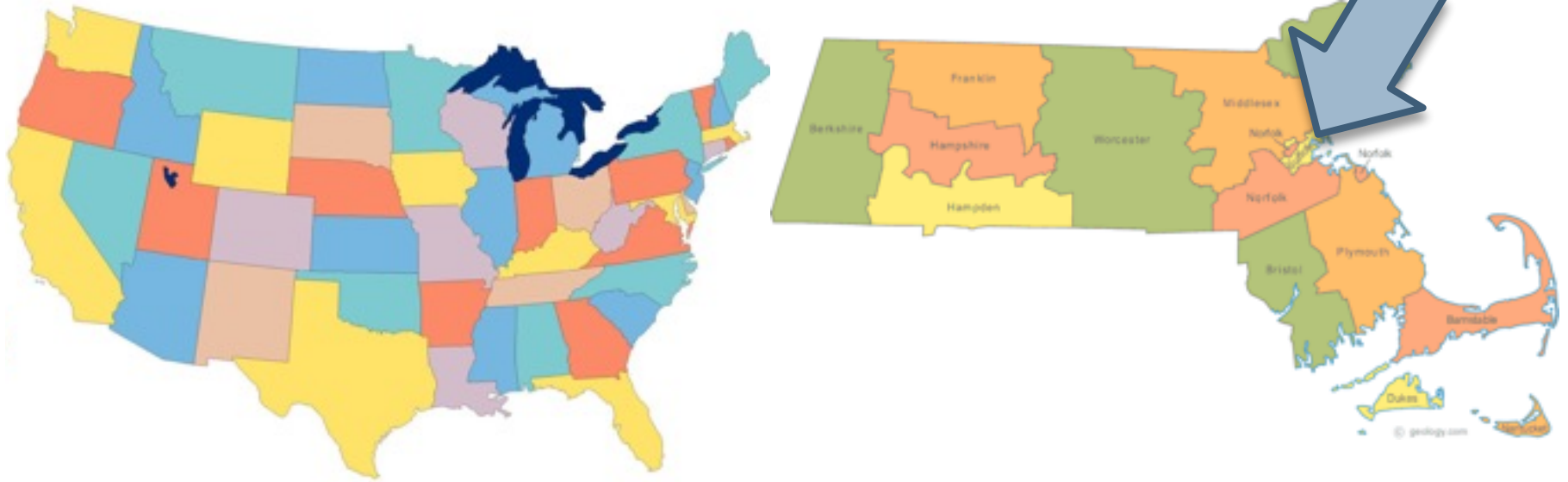
# Example: Telephone Numbers

1-617-373-1234



# Example: Telephone Numbers

1-617-373-1234



Northeastern University

# Example: Telephone Numbers

1-617-373-1234



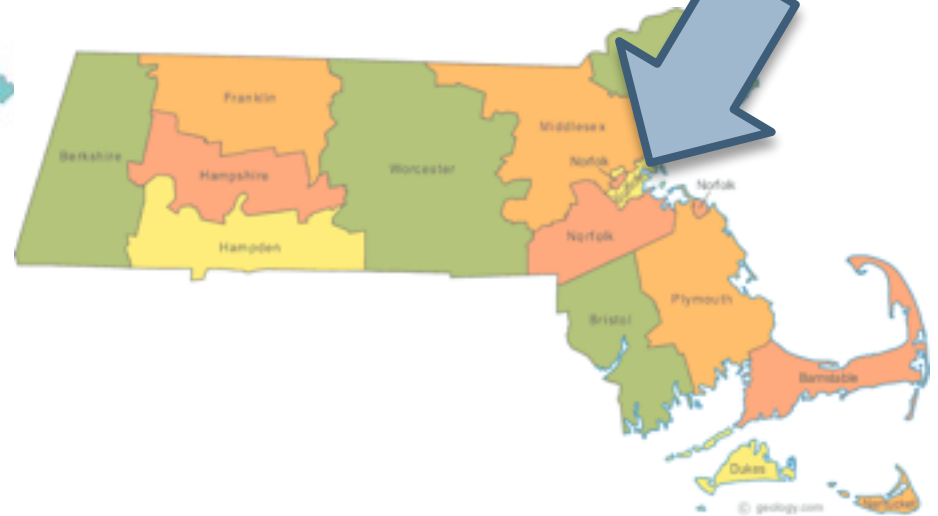
Northeastern University

West Village H  
Room 256

# Example: Telephone Numbers

1-617-373-1234

Very General



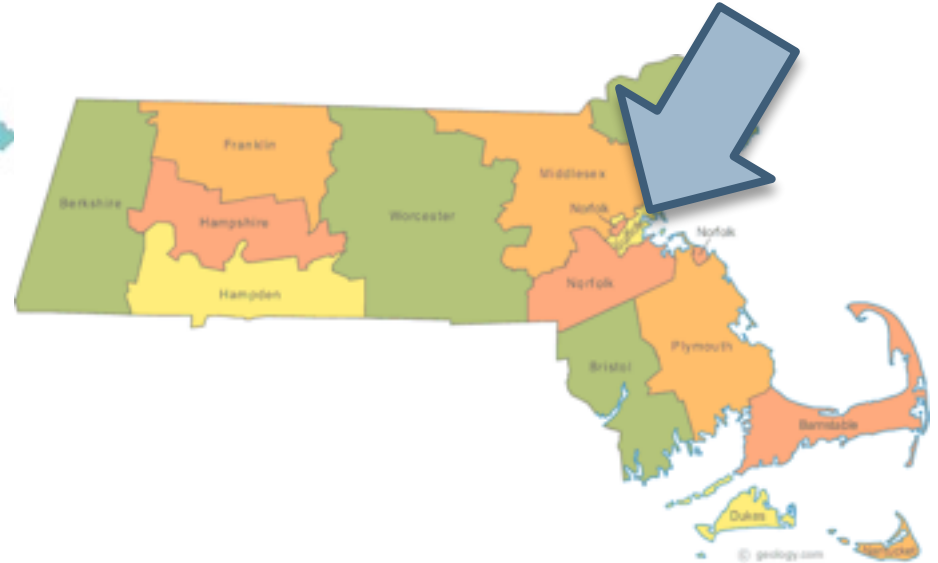
Northeastern University

West Village H  
Room 256

Very Specific

# Example: Telephone Numbers

1-617-373-1234



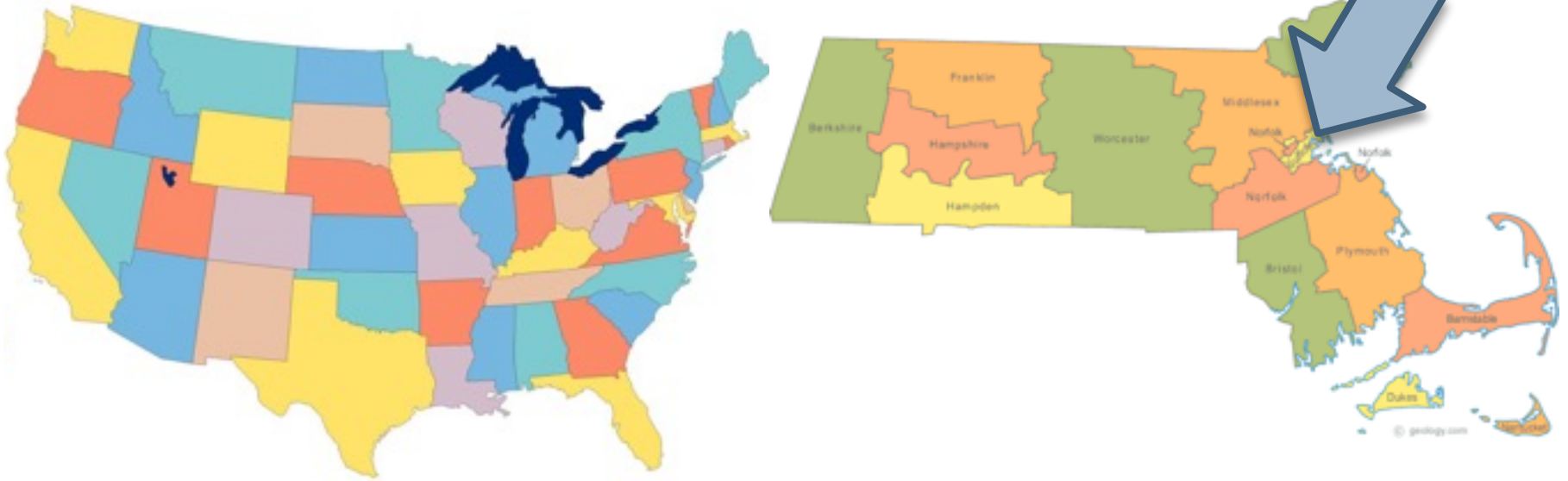
Northeastern University

West Village H  
Room 256

# Example: Telephone Numbers

---

1-617-3733278



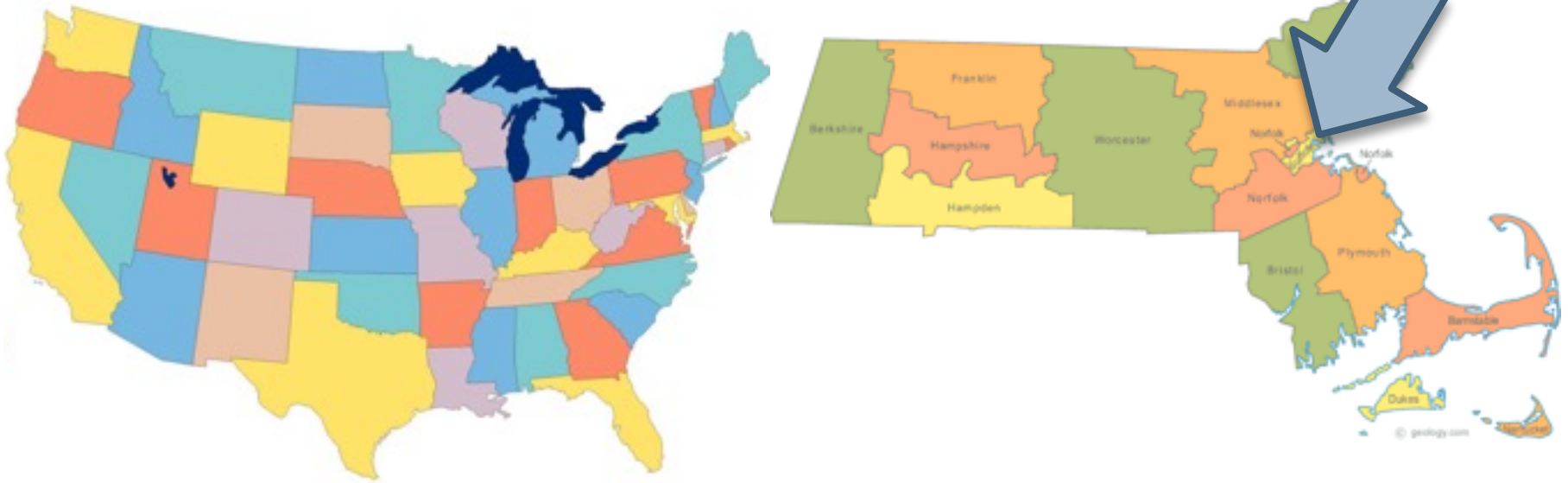
Northeastern University

West Village G  
Room 1234



# Example: Telephone Numbers

1-617-3733278



Northeastern University

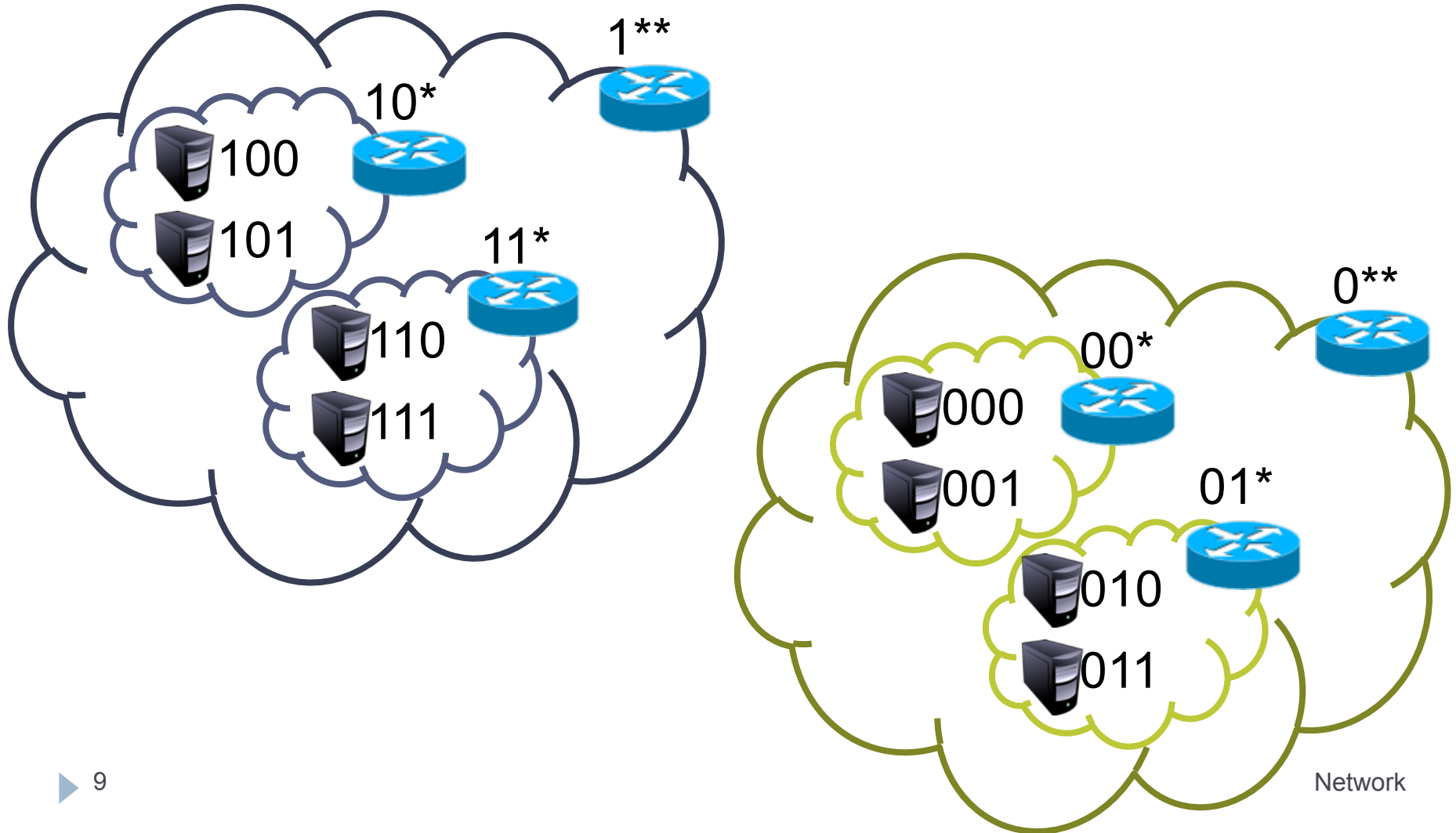
West Village G  
Room 1234

Updates are Local

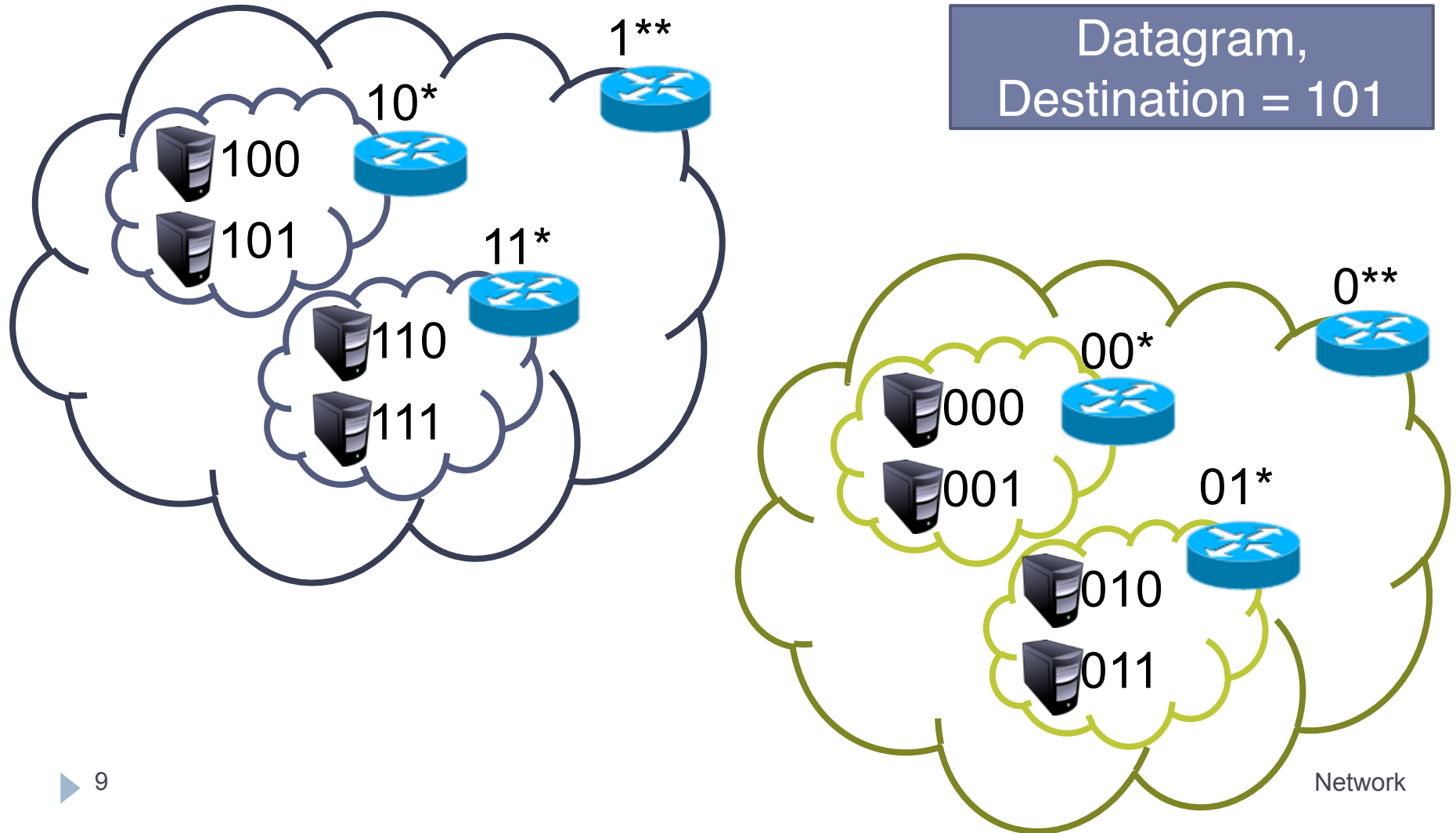
Network

# Binary Hierarchy Example

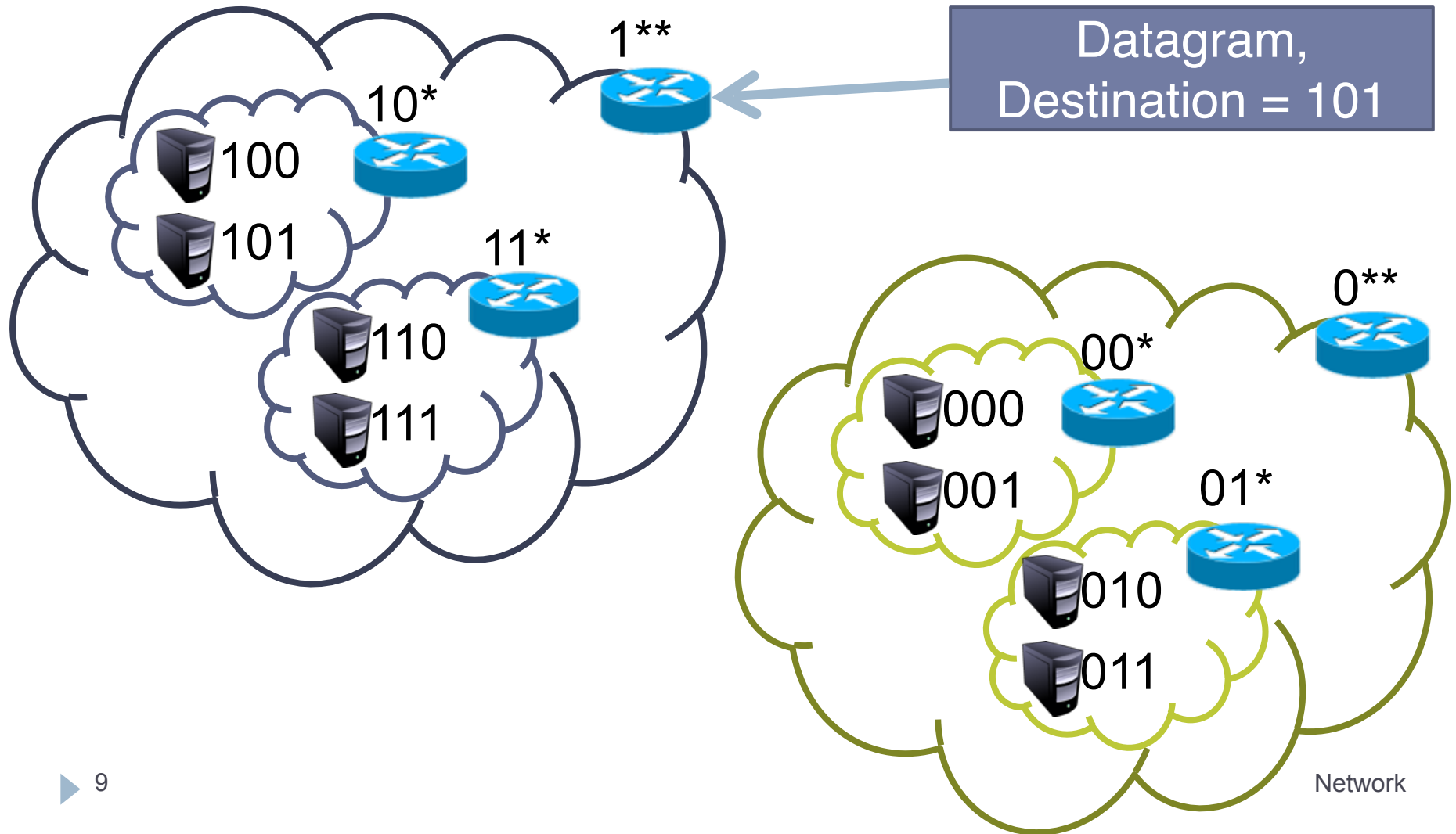
---



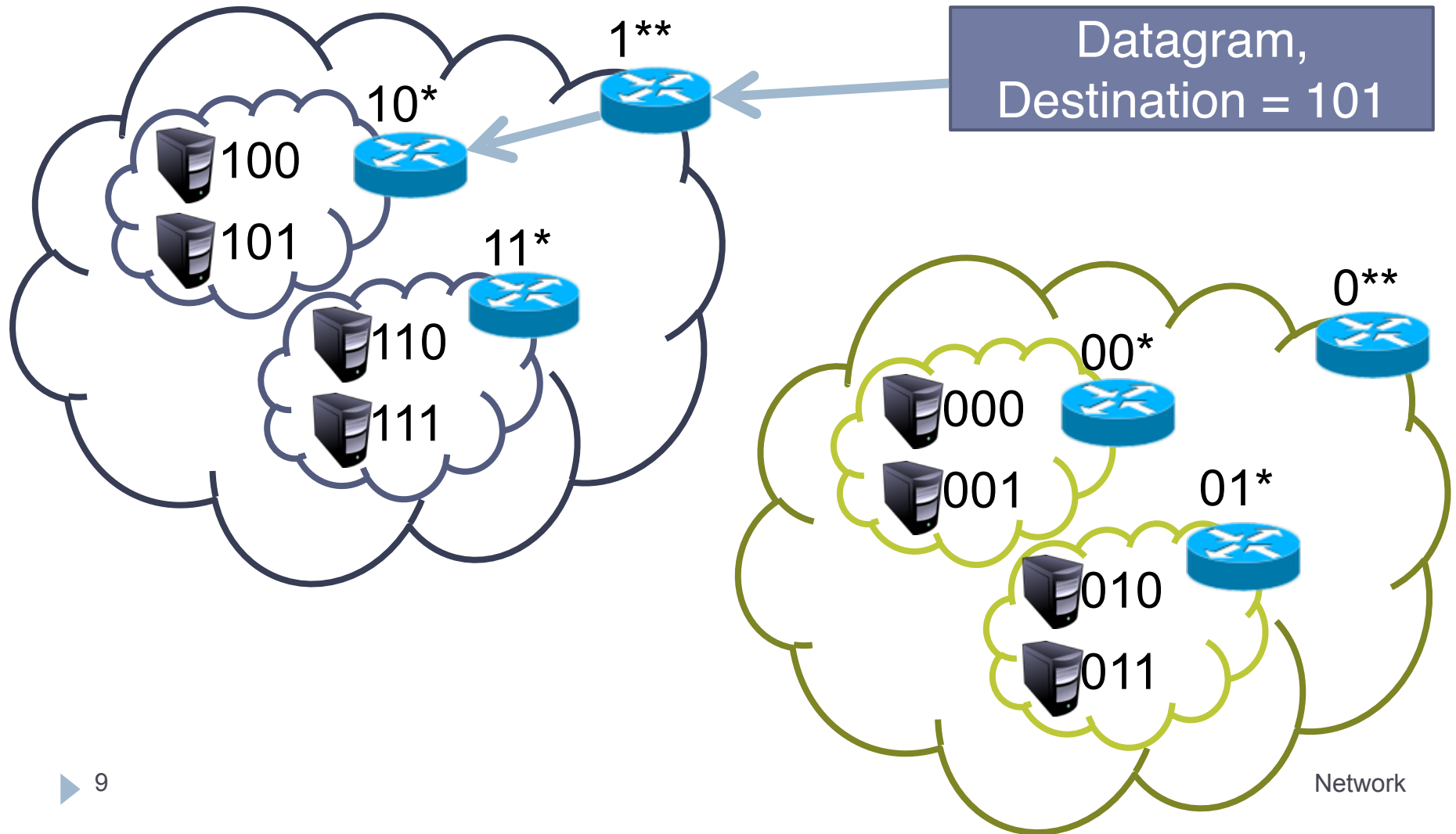
# Binary Hierarchy Example



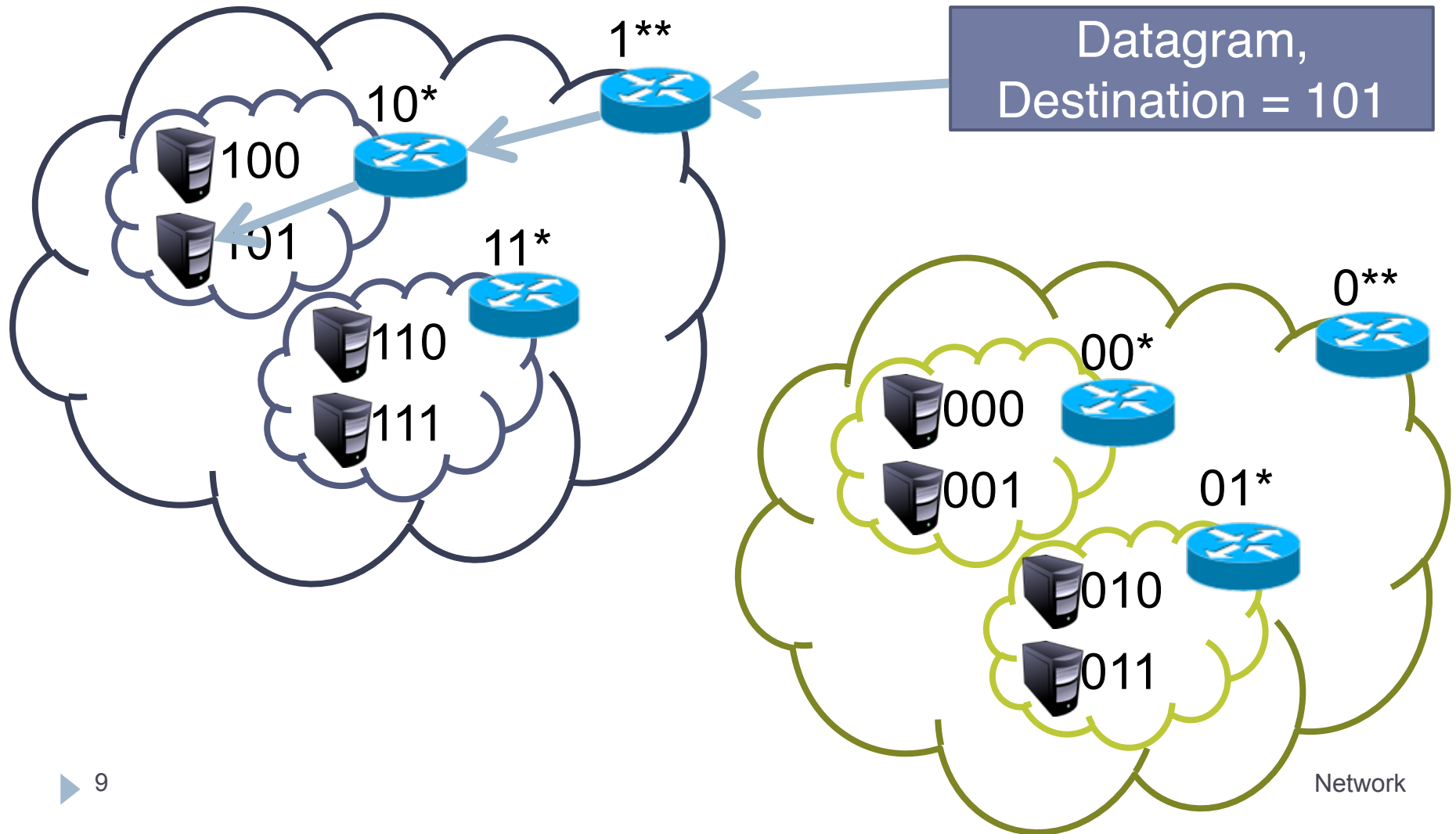
# Binary Hierarchy Example



# Binary Hierarchy Example



# Binary Hierarchy Example



# IP Addressing

---

- ▶ **IPv4: 32-bit addresses**

- ▶ Usually written in dotted notation, e.g. 192.168.21.76
- ▶ Each number is a byte
- ▶ Stored in Big Endian order

	0	8	16	24	31
Decimal	192	168	21	76	
Hex	C0	A8	15	4C	
Binary	11000000	10101000	00010101	01001100	

# IP Addressing and Forwarding

---

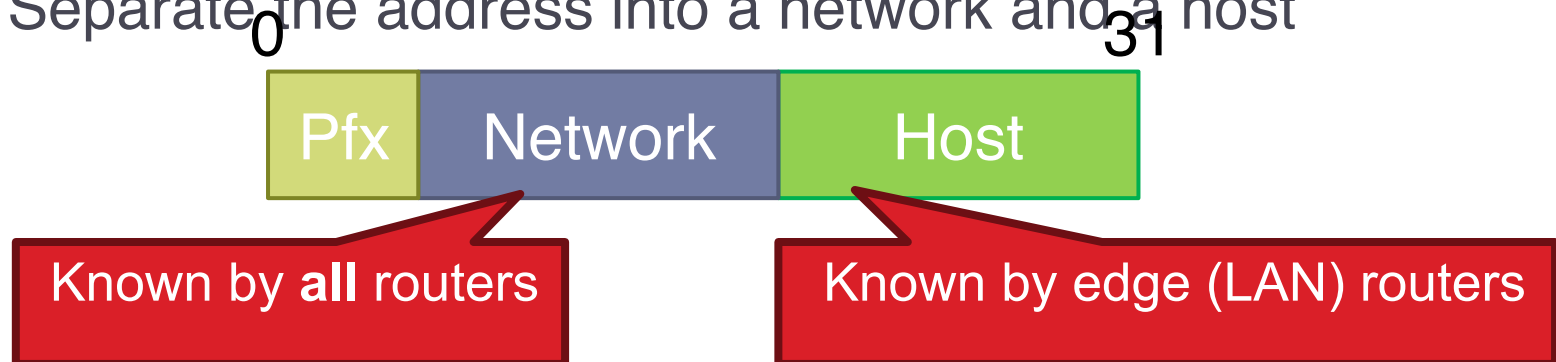
## ▶ Routing Table Requirements

- ▶ For every possible IP, give the next hop
- ▶ But for 32-bit addresses,  $2^{32}$  possibilities!
- ▶ **Too slow**: 48GE ports and 4x10GE needs 176Gbps bandwidth

**DRAM**: ~1-6 Gbps; **TCAM** is fast, but 400x cost of DRAM

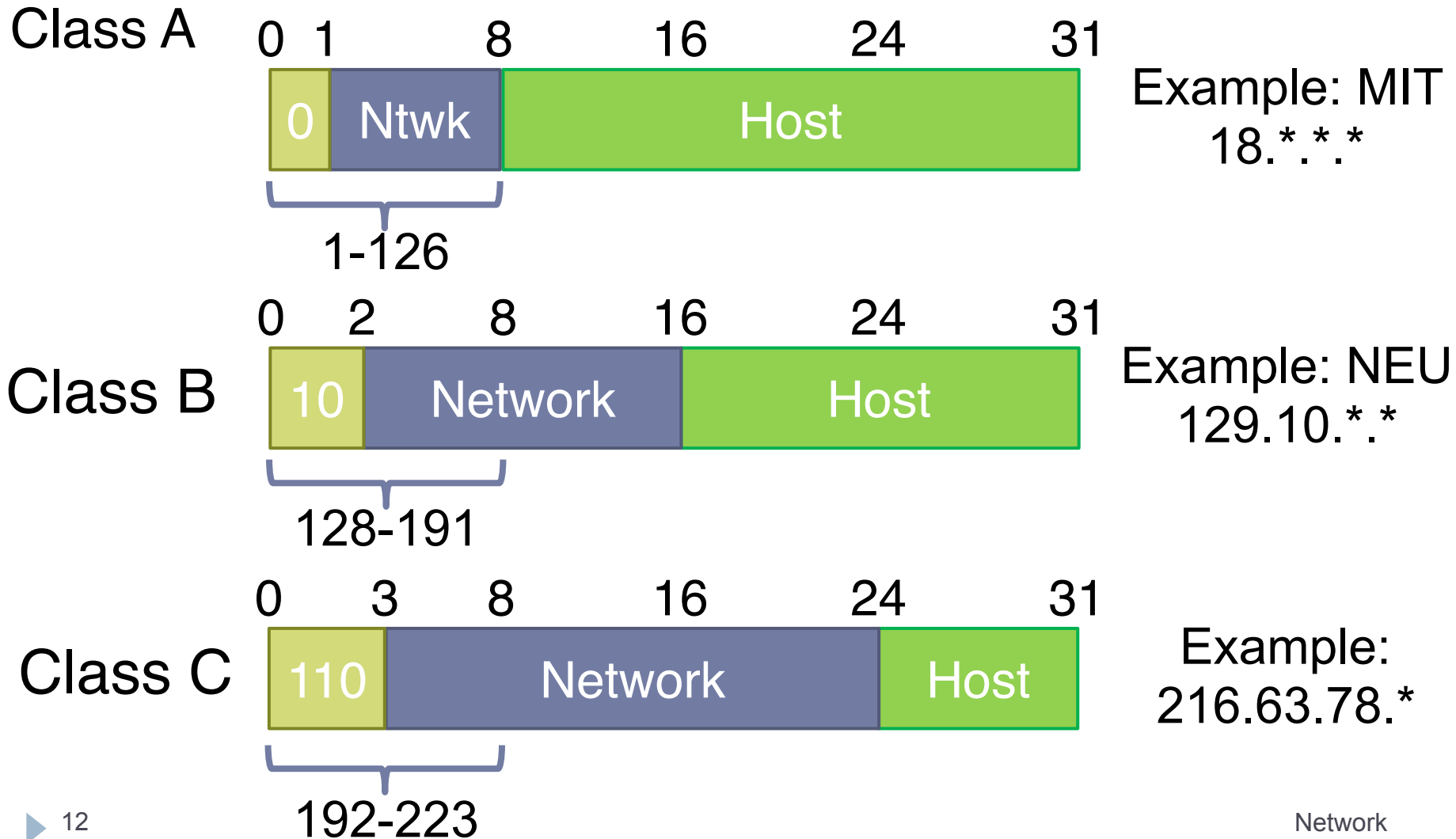
## ▶ Hierarchical address scheme

- ▶ Separate the address into a network and a host





# Classes of IP Addresses



# How Do You Get IPs?

---

- ▶ IP address ranges controlled by IANA

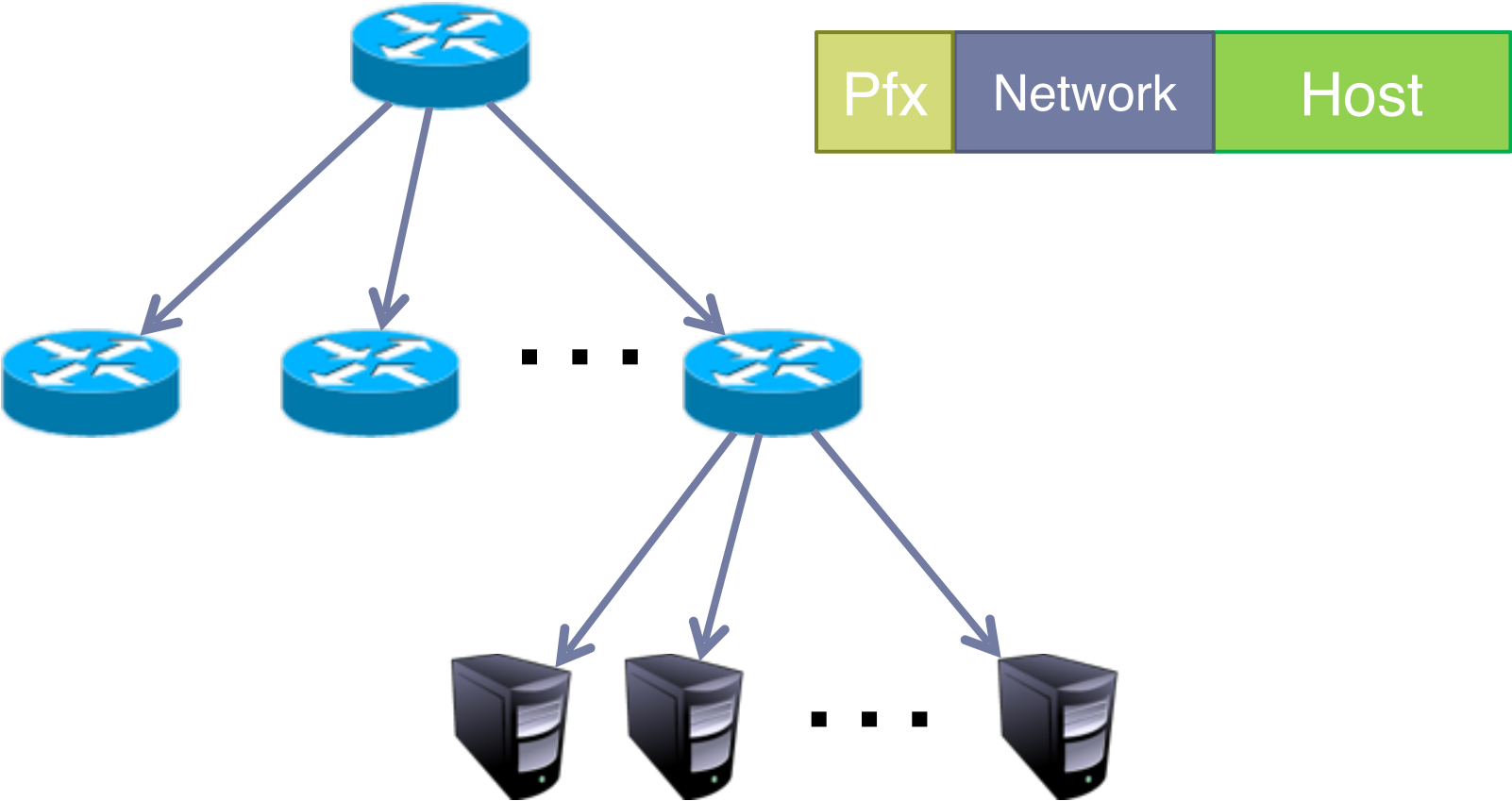


Internet Assigned Numbers Authority

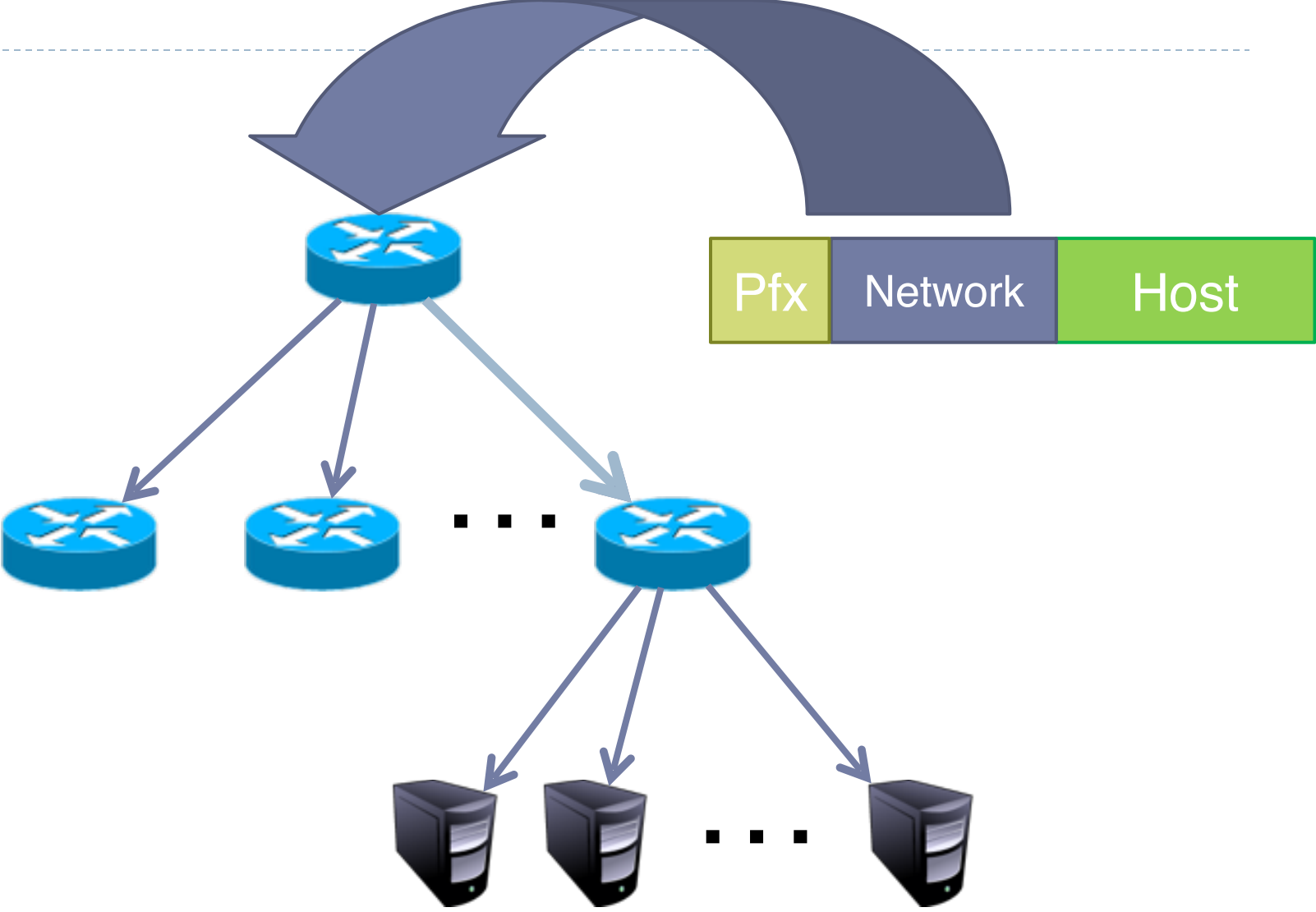
- ▶ Internet Assigned Number Authority
- ▶ Roots go back to 1972, ARPANET, UCLA
- ▶ Today, part of ICANN
- ▶ **IANA grants IPs to regional authorities**
  - ▶ ARIN (American Registry of Internet Numbers) may grant you a range of IPs
  - ▶ You may then advertise routes to your new IP range
  - ▶ There are now secondary markets, auctions, ...

# Two Level Hierarchy

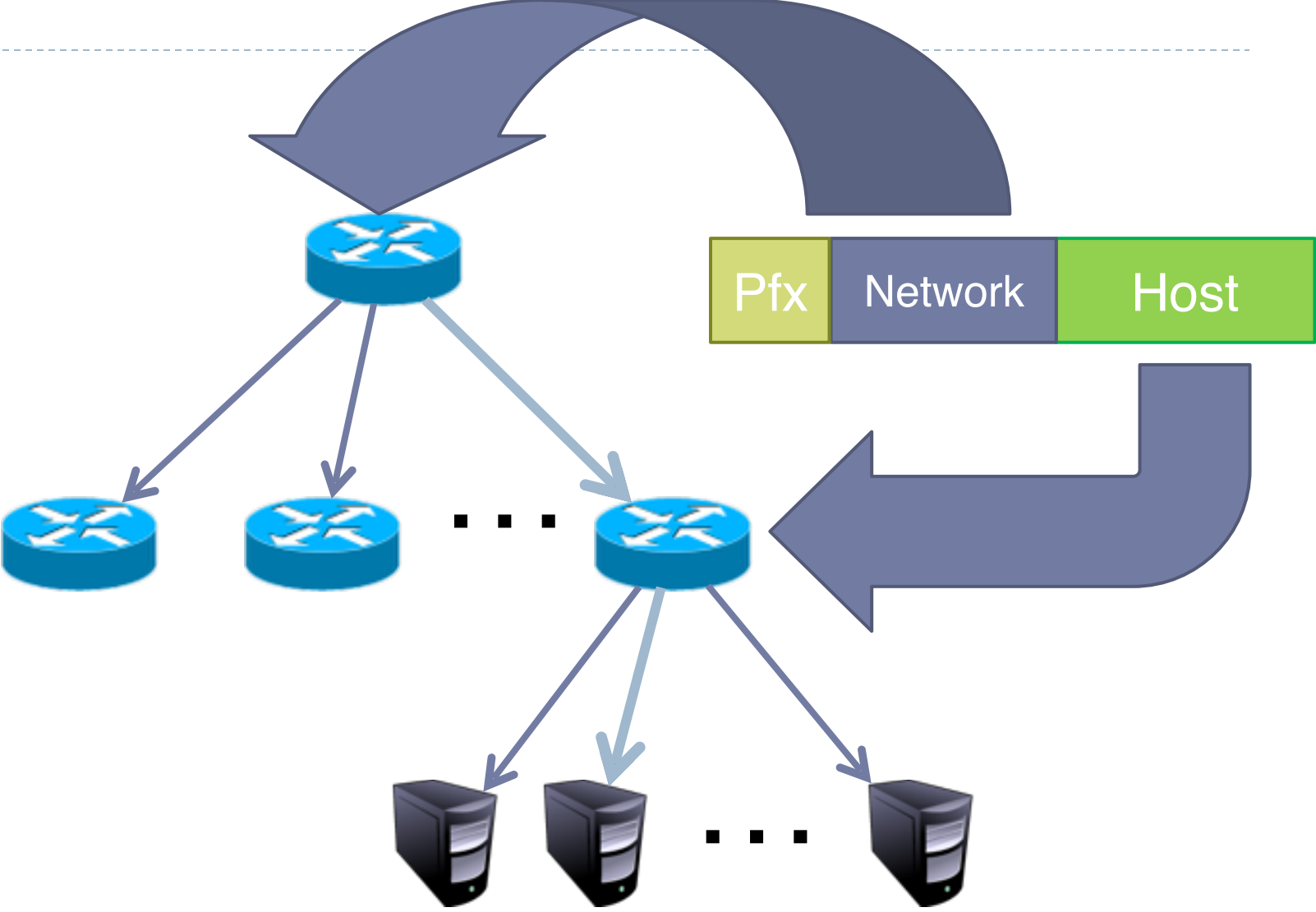
---



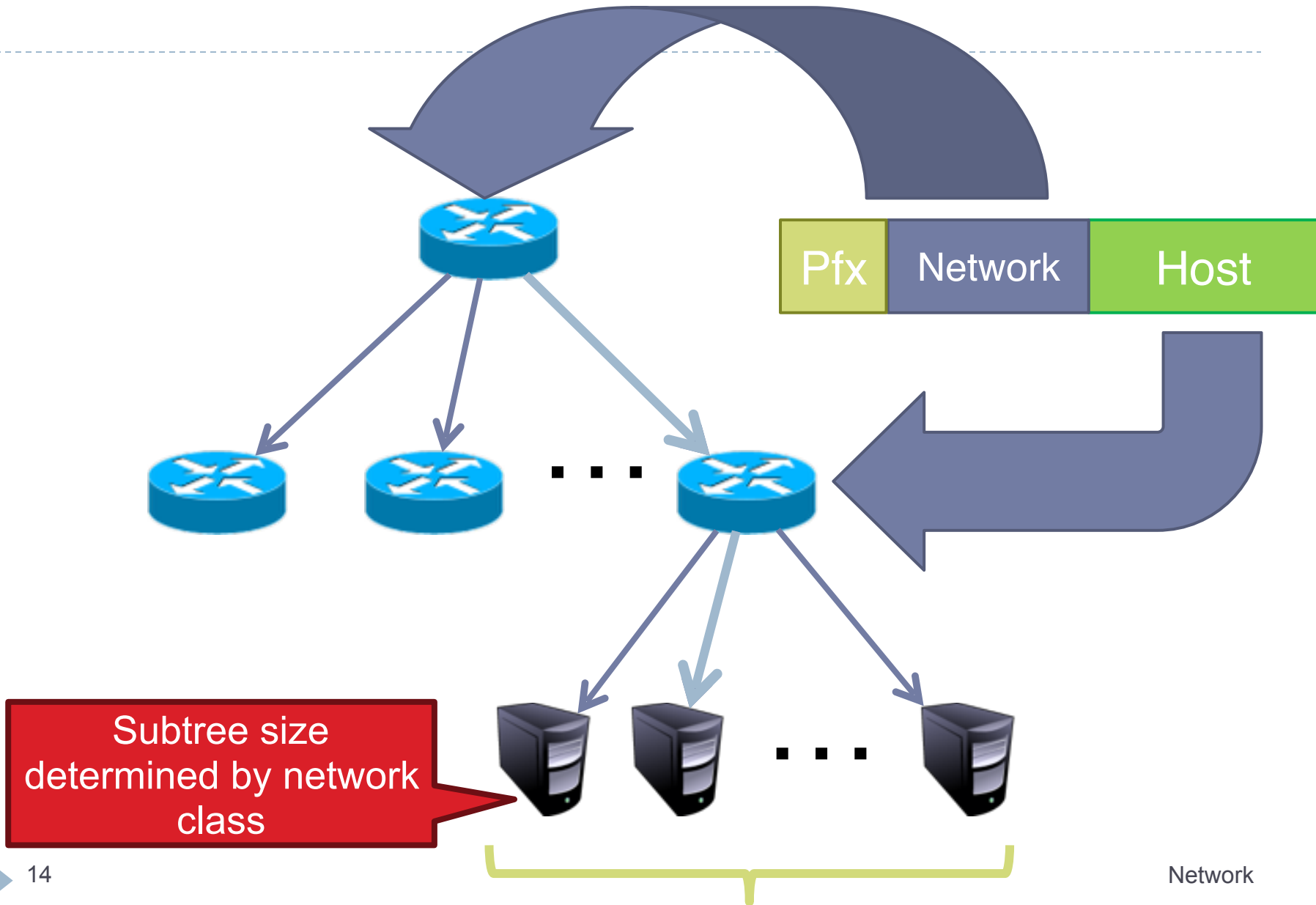
# Two Level Hierarchy



# Two Level Hierarchy



# Two Level Hierarchy



# Class Sizes

Way too big

Class	Prefix Bits	Network Bits	Number of Classes	Hosts per Class
A	1	7	$2^7 - 2 = 126$ (0 and 127 are reserved)	$2^{24} - 2 = 16,777,214$ (All 0 and all 1 are reserved)
B	2	14	$2^{14} = 16,398$	$2^{16} - 2 = 65,534$ (All 0 and all 1 are reserved)
C	3	21	$2^{21} = 2,097,512$	$2^8 - 2 = 254$ (All 0 and all 1 are reserved)
			Total: 2,114,036	

Too many  
network IDs

Too small to  
be useful

# Subnets

---

- ▶ Problem: need to break up large A and B classes
- ▶ Solution: add another layer to the hierarchy
  - ▶ From the outside, appears to be a single network
    - ▶ Only 1 entry in routing tables
  - ▶ Internally, manage multiple subnetworks
    - ▶ Split the address range using a **subnet mask**



Subnet Mask: 11111111 11111111 11000000 00000000



# Subnet Example

---

## Extract network:

IP Address: 10110101 11011101 01010100 01110010

Subnet Mask: & 11111111 11111111 11000000 00000000

Result: 10110101 11011101 01000000 00000000

## Extract host:

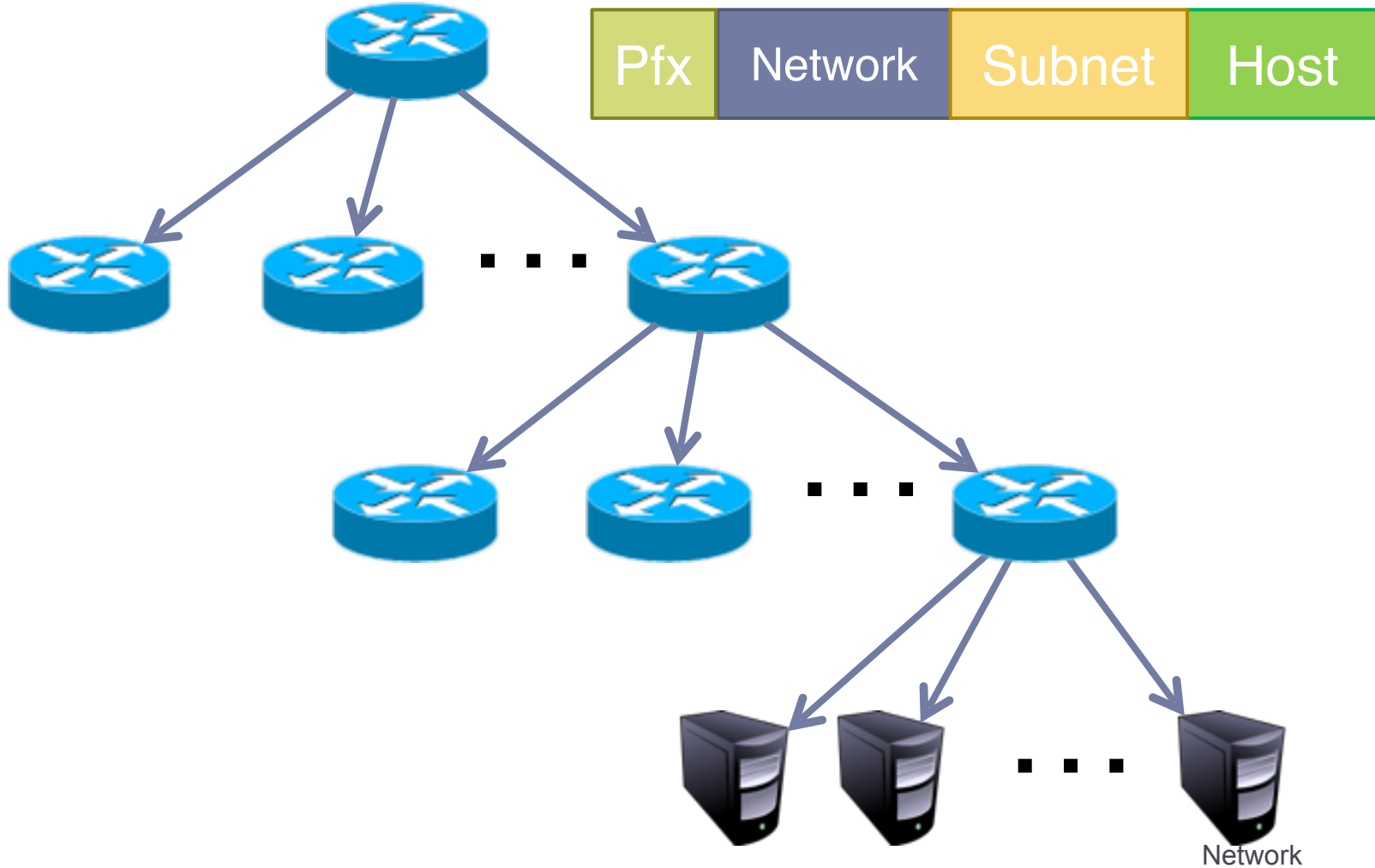
IP Address: 10110101 11011101 01010100 01110010

Subnet Mask: & ~(11111111 11111111 11000000 00000000)

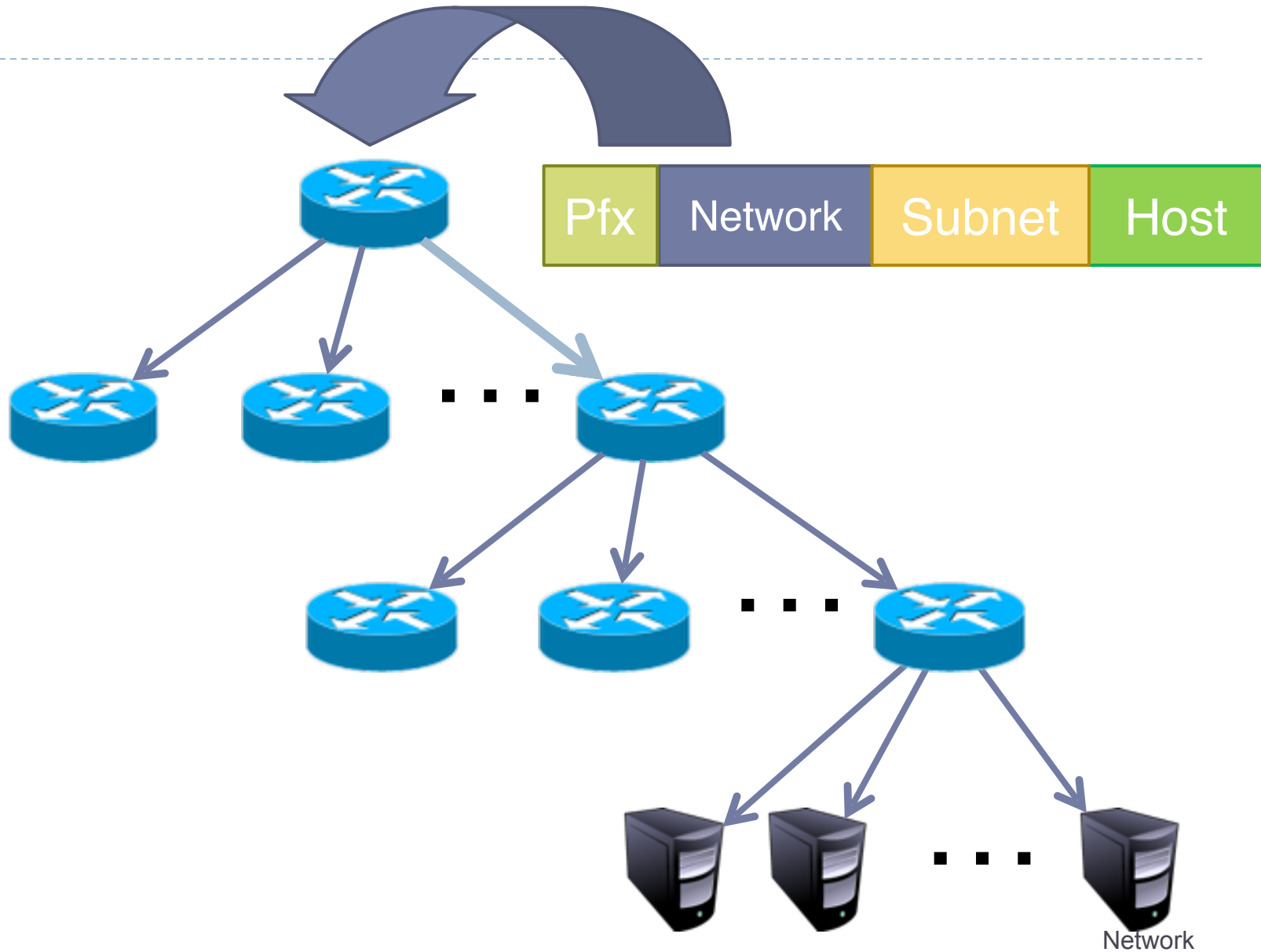
Result: 00000000 00000000 00010100 01110010

# N-Level Subnet Hierarchy

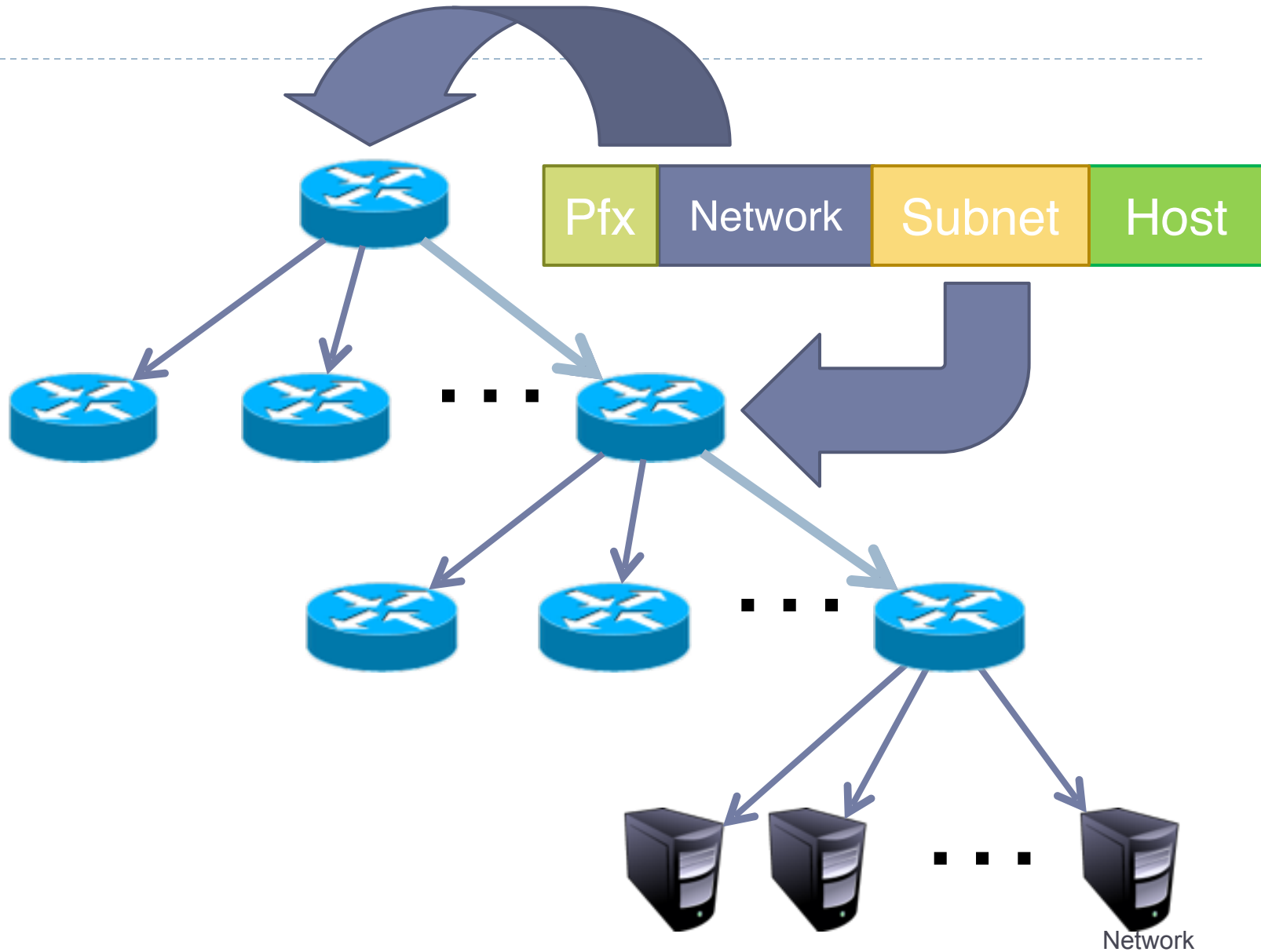
---



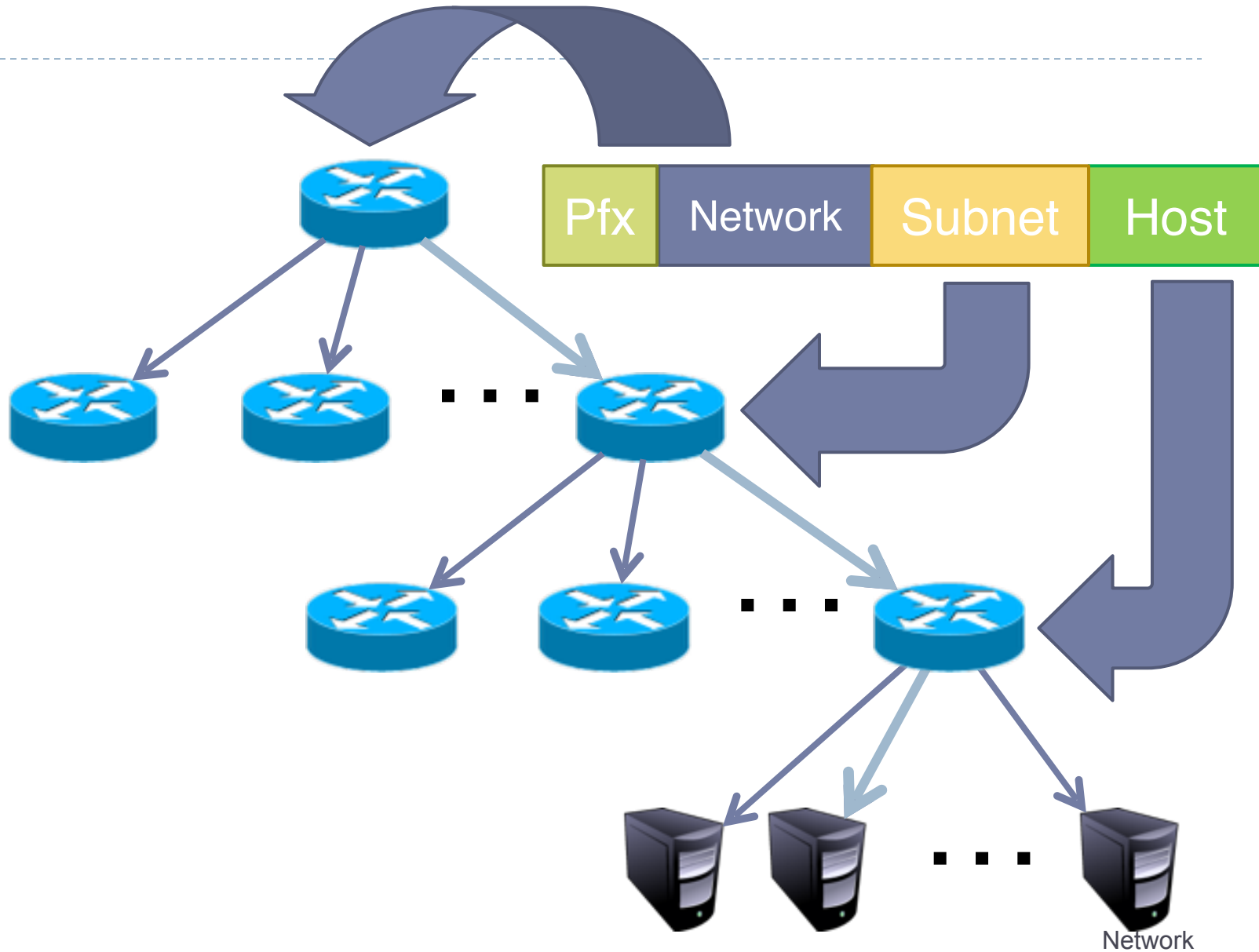
# N-Level Subnet Hierarchy



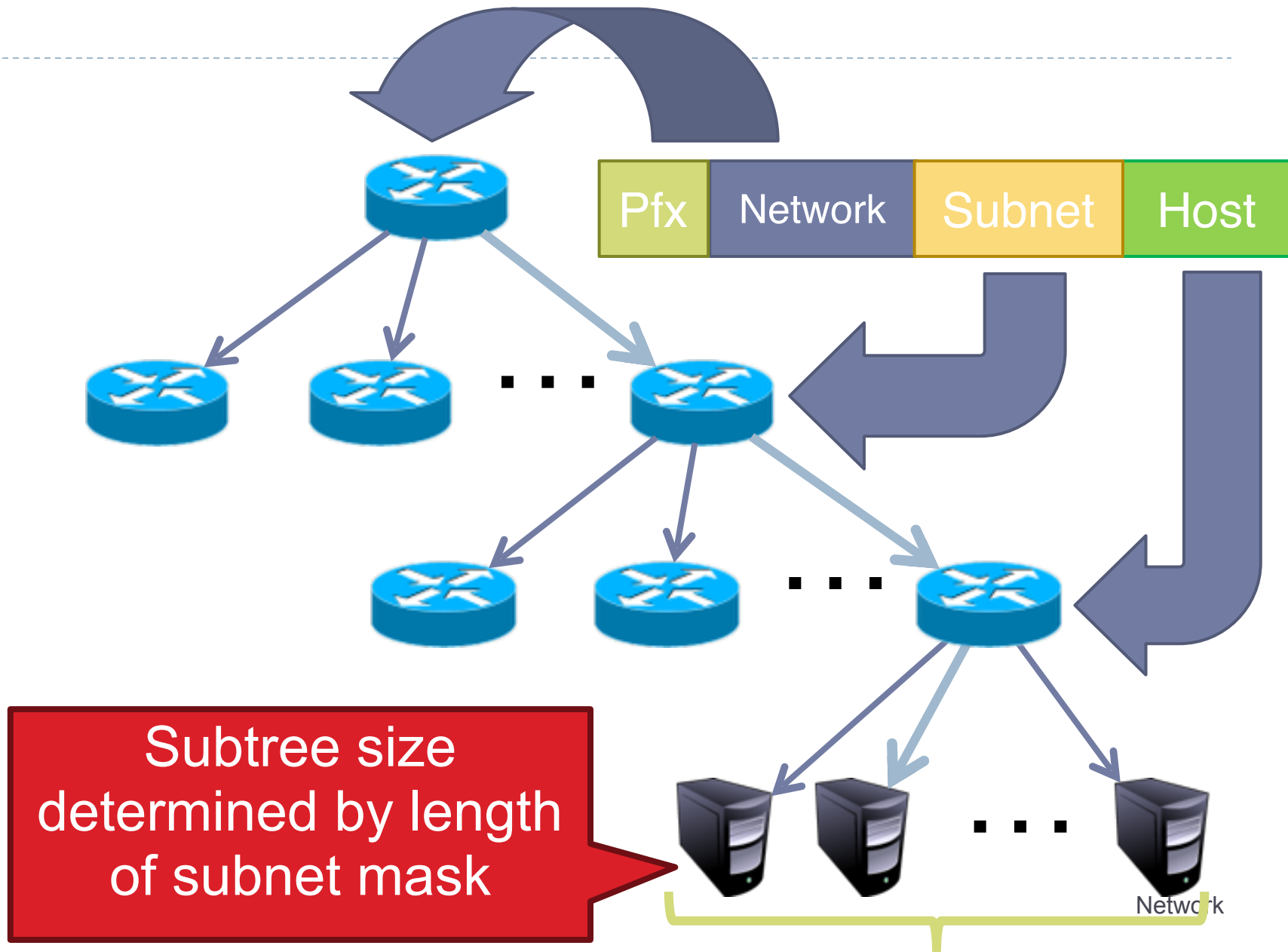
# N-Level Subnet Hierarchy



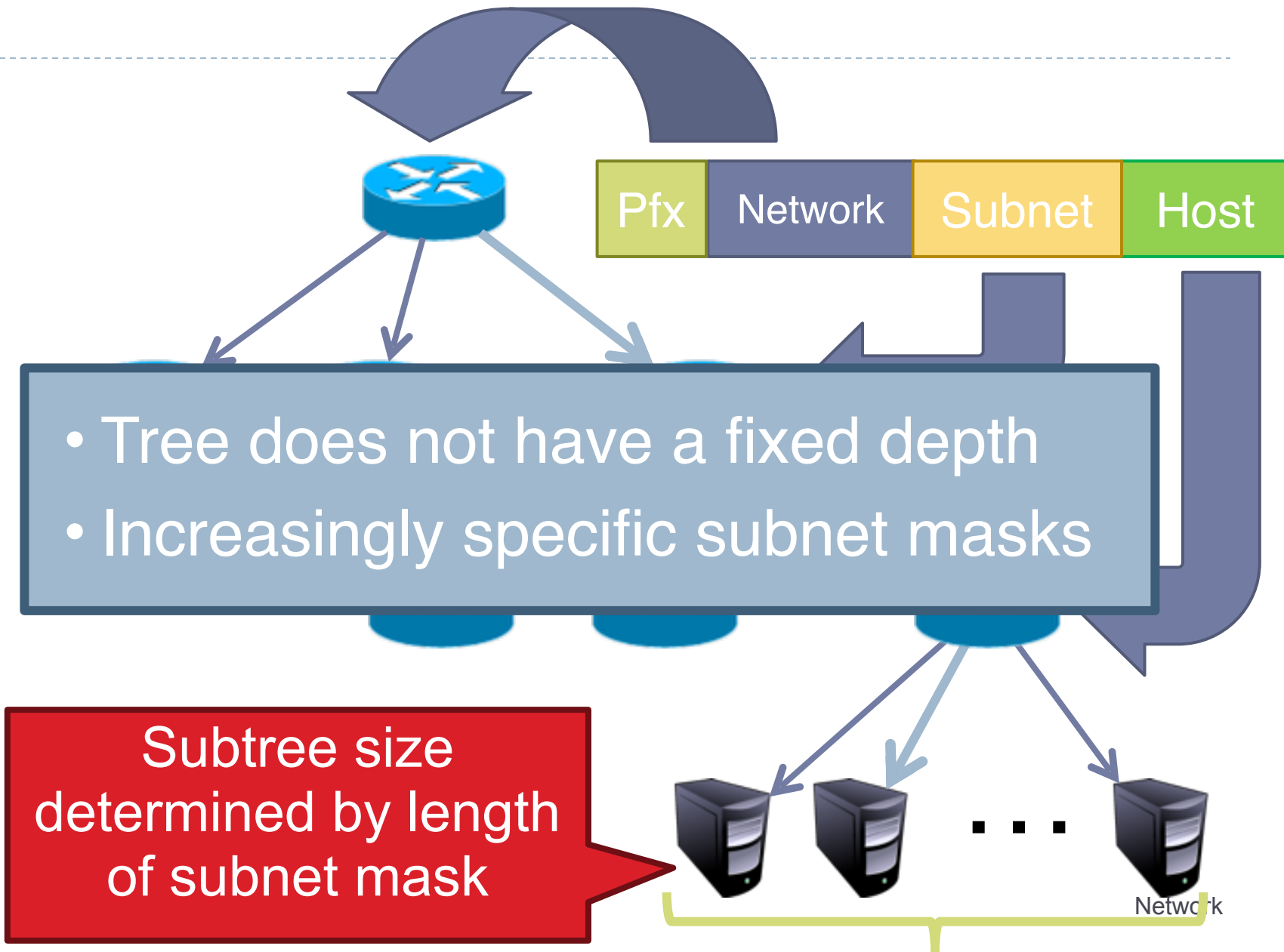
# N-Level Subnet Hierarchy



# N-Level Subnet Hierarchy



# N-Level Subnet Hierarchy



# Example Routing Table

---

Address Pattern	Subnet Mask	Destination Router
0.0.0.0	0.0.0.0	Router 4
18.0.0.0	255.0.0.0	Router 2
128.42.0.0	255.255.0.0	Router 3
128.42.128.0	255.255.128.0	Router 5
128.42.222.0	255.255.255.0	Router 1

- Question: 128.42.222.198 matches four rows
  - ▣ Which router do we forward to?
- **Longest prefix matching**
  - ▣ Use the row with the longest number of 1's in the mask
  - ▣ This is the **most specific match**



# Subnetting Revisited

---

- ▶ Question: does subnetting solve all the problems of class-based routing?

NO

- ▶ **Classes are still too coarse**
  - ▶ Class A can be subnetted, but only 126 available
  - ▶ Class C is too small
  - ▶ Class B is nice, but there are only 16,398 available
- ▶ **Routing tables are still too big**
  - ▶ 2.1 million entries per router

# Classless Inter Domain Routing (CIDR)

---

- ▶ CIDR, pronounced 'cider'
- ▶ Key ideas:
  - ▶ Get rid of IP classes
  - ▶ Use bitmasks for all levels of routing
  - ▶ **Aggregation** to minimize FIB (forwarding information base)

# Classless Inter Domain Routing (CIDR)

---

- ▶ CIDR, pronounced 'cider'
- ▶ Key ideas:
  - ▶ Get rid of IP classes
  - ▶ Use bitmasks for all levels of routing
  - ▶ **Aggregation** to minimize FIB (forwarding information base)
- ▶ Arbitrary split between network and host
  - ▶ Specified as a bitmask or prefix length
  - ▶ Example: Northeastern
    - ▶ 129.10.0.0 with `netmask 255.255.0.0`
    - ▶ 129.10.0.0 / 16

# Aggregation with CIDR

---

- ▶ Original use: aggregating class C ranges
- ▶ One organization given contiguous class C ranges
  - ▶ Example: Microsoft, 207.46.192.\* – 207.46.255.\*
  - ▶ Represents  $2^6 = 64$  class C ranges
  - ▶ Specified as CIDR address 207.46.192.0/18

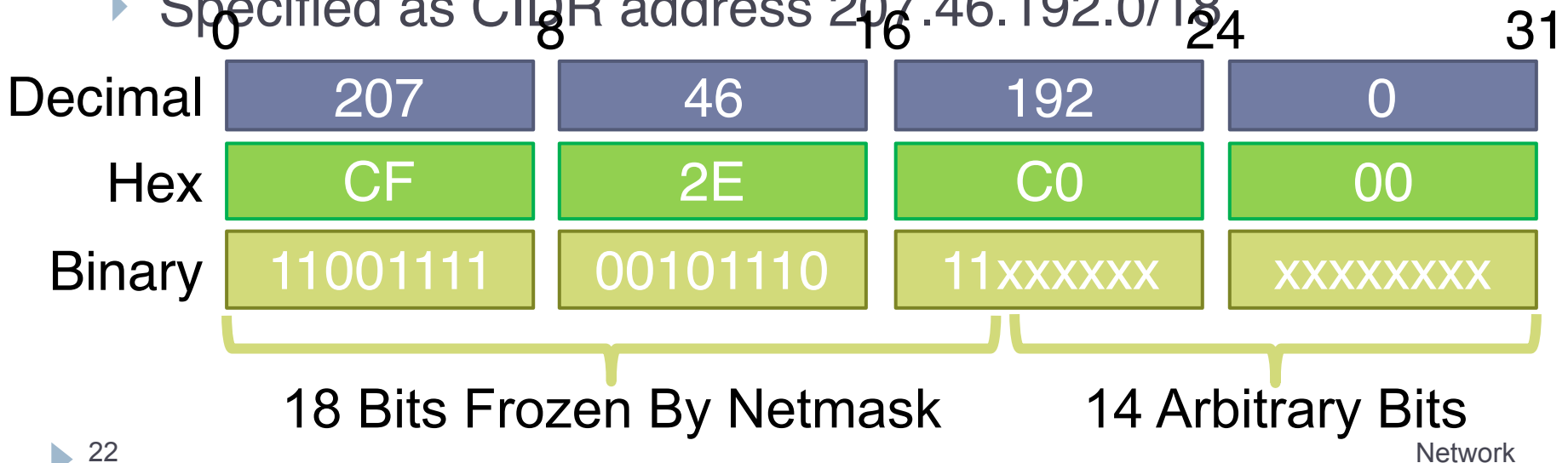
# Aggregation with CIDR

- ▶ Original use: aggregating class C ranges
- ▶ One organization given contiguous class C ranges
  - ▶ Example: Microsoft, 207.46.192.\* – 207.46.255.\*
  - ▶ Represents  $2^6 = 64$  class C ranges
  - ▶ Specified as CIDR address 207.46.192.0/18

	0	8	16	24	31
Decimal	207	46	192	0	
Hex	CF	2E	C0	00	
Binary	11001111	00101110	11xxxxxx	xxxxxxxx	

# Aggregation with CIDR

- ▶ Original use: aggregating class C ranges
- ▶ One organization given contiguous class C ranges
  - ▶ Example: Microsoft, 207.46.192.\* – 207.46.255.\*
  - ▶ Represents  $2^6 = 64$  class C ranges
  - ▶ Specified as CIDR address 207.46.192.0/18



# Example CIDR Routing Table

---

Address	Netmask	Third Byte	Byte Range
207.46.0.0	19	000xxxxx	0 – 31
207.46.32.0	19	001xxxxx	32 – 63
207.46.64.0	19	010xxxxx	64 – 95
207.46.128.0	18	10xxxxxx	128 – 191
207.46.192.0	18	11xxxxxx	192 – 255

# Example CIDR Routing Table

---

Address	Netmask	Third Byte	Byte Range
207.46.0.0	19	000xxxxx	0 – 31
207.46.32.0	19	001xxxxx	32 – 63
207.46.64.0	19	010xxxxx	64 – 95
207.46.128.0	18	10xxxxxx	128 – 191
207.46.192.0	18	11xxxxxx	192 – 255



# Example CIDR Routing Table

---

Address	Netmask	Third Byte	Byte Range
207.46.0.0	19	000xxxxx	0 – 31
207.46.32.0	19	001xxxxx	32 – 63
207.46.64.0	19	010xxxxx	64 – 95
207.46.128.0	18	10xxxxxx	128 – 191
207.46.192.0	18	11xxxxxx	192 – 255

# Example CIDR Routing Table

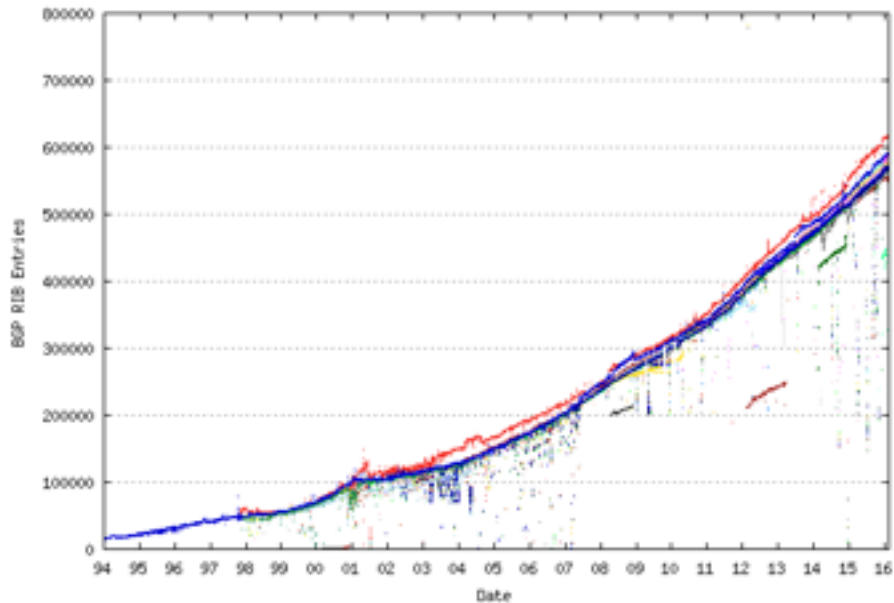
Address	Netmask	Third Byte	Byte Range
207.46.0.0	19	000xxxxx	0 – 31
207.46.32.0	19	001xxxxx	32 – 63
207.46.64.0	19	010xxxxx	64 – 95
207.46.128.0	18	10xxxxxx	128 – 191
207.46.192.0	18	11xxxxxx	192 – 255

Hole in the Routing Table: No coverage for 96 – 127  
207.46.96.0/19

# Size of CIDR Routing Tables

---

- ▶ From [www.cidr-report.org](http://www.cidr-report.org)
- ▶ CIDR has kept IP routing table sizes in check
  - ▶ Currently ~600,000 entries for a complete IP routing table (up 33% from '14)
  - ▶ Only required by backbone routers



# Takeaways

---

- ▶ **Hierarchical addressing is critical for scalability**
  - ▶ Not all routers need all information
  - ▶ Limited number of routers need to know about changes
- ▶ **Non-uniform hierarchy useful for heterogeneous networks**
  - ▶ Class-based addressing is too coarse
  - ▶ CIDR improves scalability and granularity
- ▶ **Implementation challenges**
  - ▶ Longest prefix matching is more difficult than schemes with no ambiguity



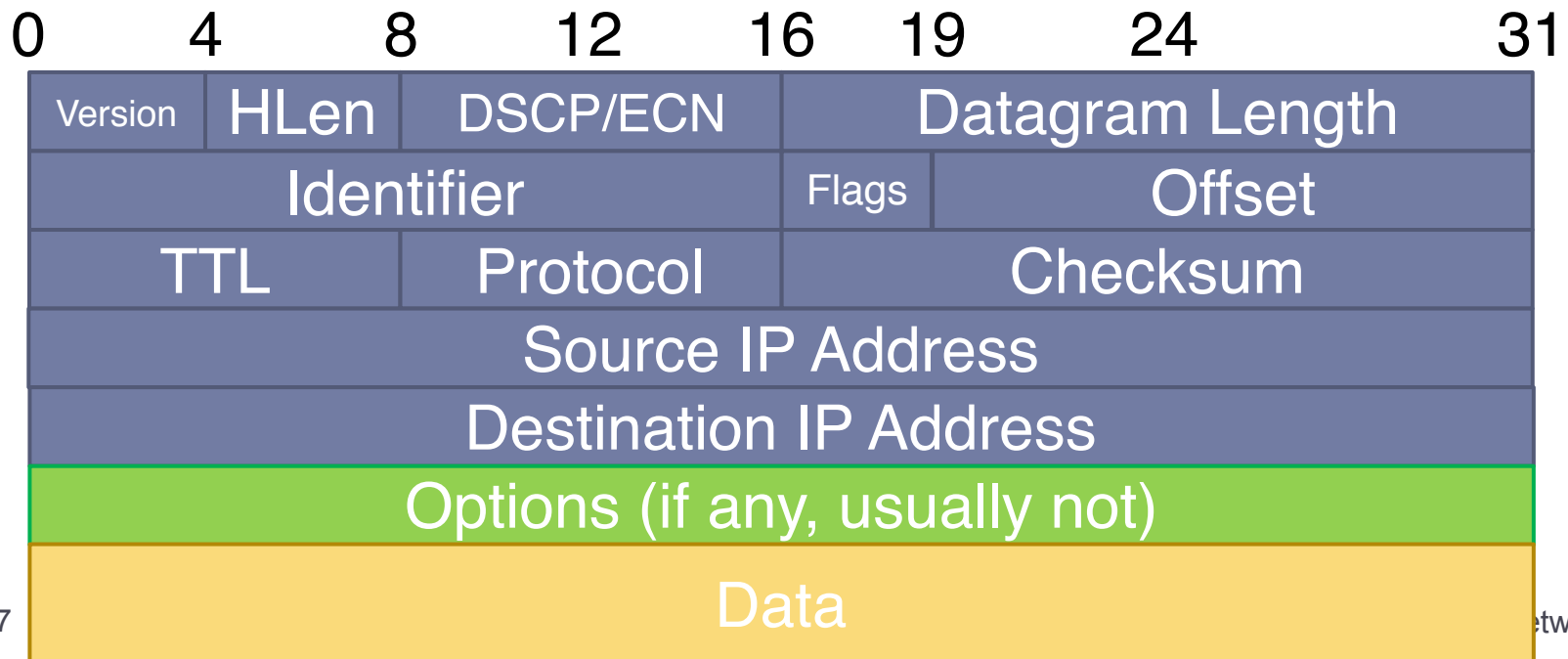


2: IPv4

# IP Datagrams

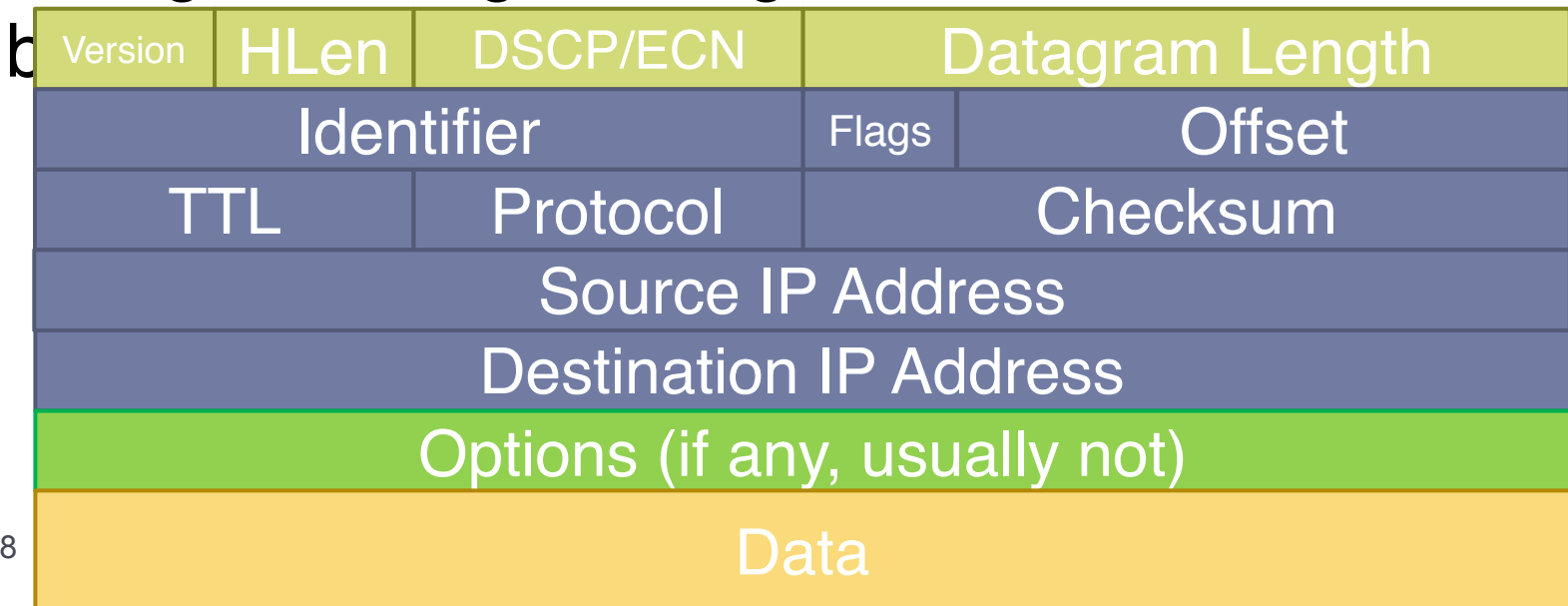
---

- ▶ IP Datagrams are like a letter
  - ▶ Totally self-contained
  - ▶ Include all necessary addressing information
  - ▶ No advanced setup of connections or circuits



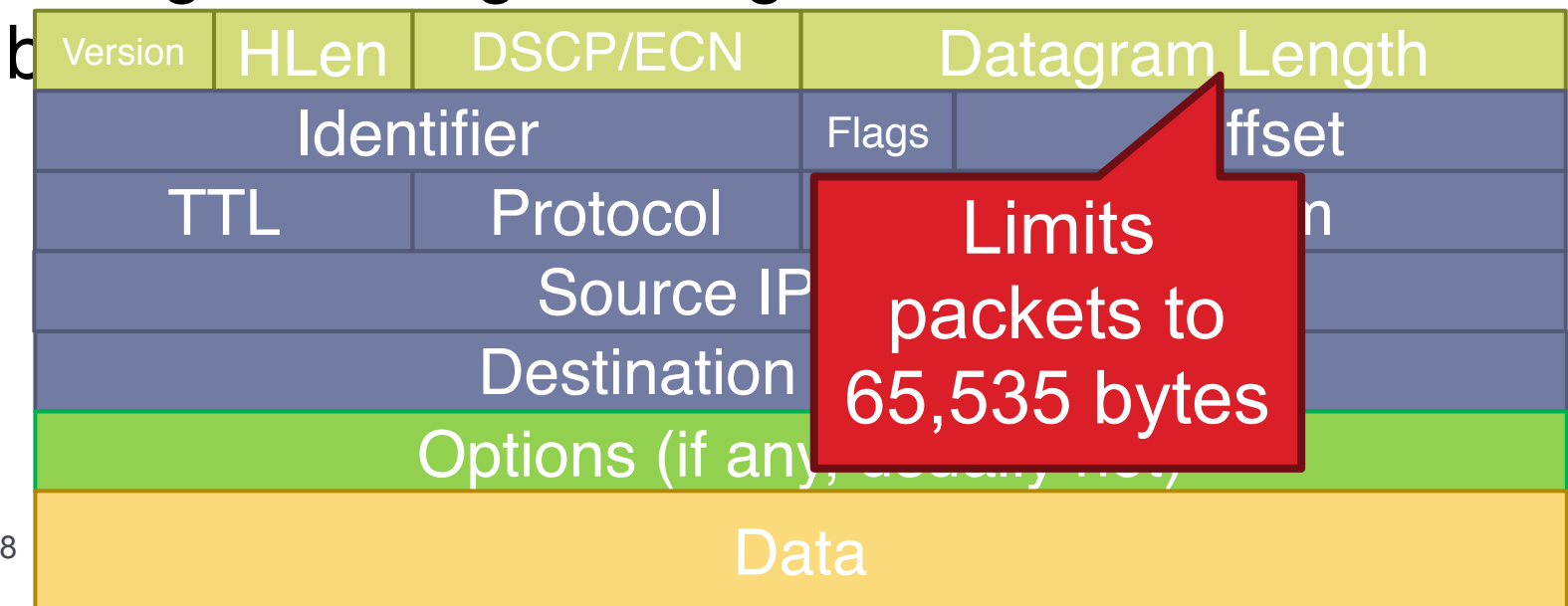
# IP Header Fields: Word 1

- ▶ Version: 4 for IPv4
- ▶ Header Length: Number of 32-bit words (usually 5)
- ▶ Type of Service: Priority information (unused)
- ▶ Datagram Length: Length of header + data in 32-bit words



# IP Header Fields: Word 1

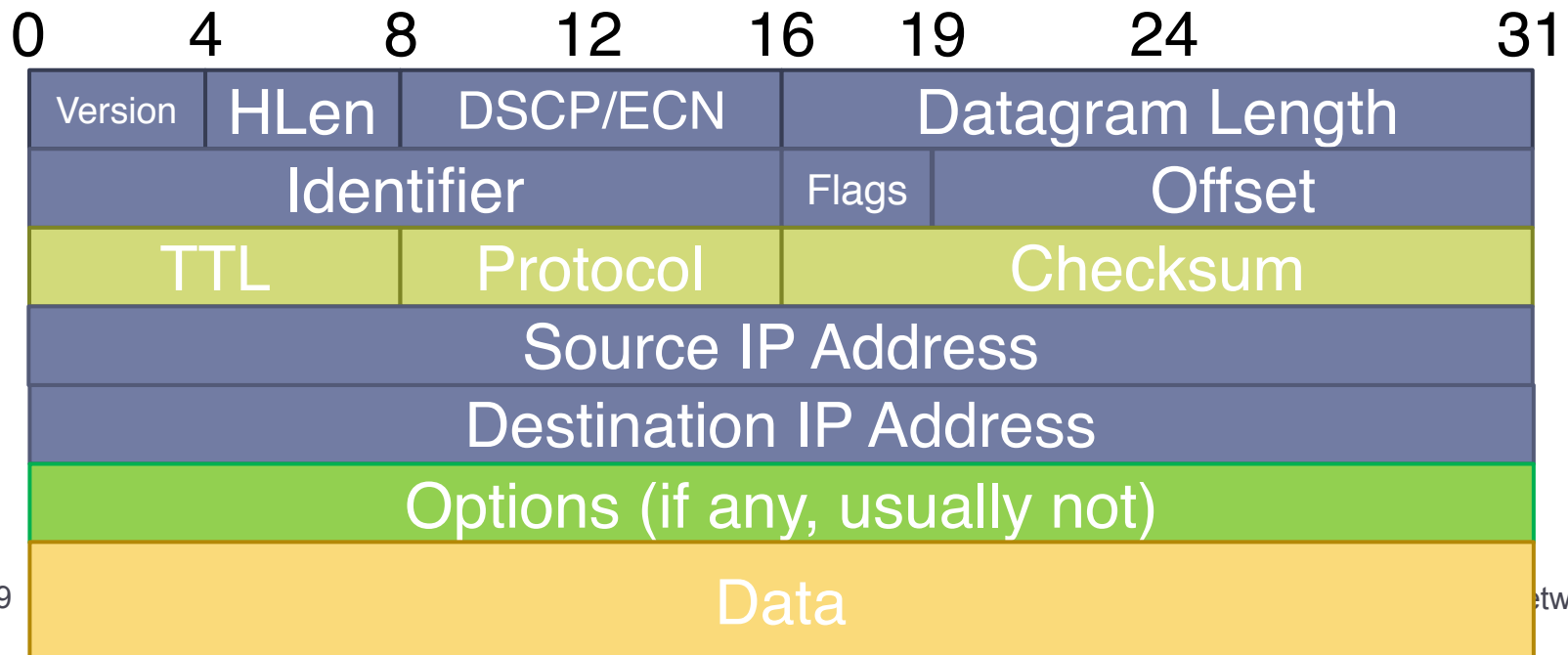
- ▶ Version: 4 for IPv4
- ▶ Header Length: Number of 32-bit words (usually 5)
- ▶ Type of Service: Priority information (unused)
- ▶ Datagram Length: Length of header + data in 32-bit words





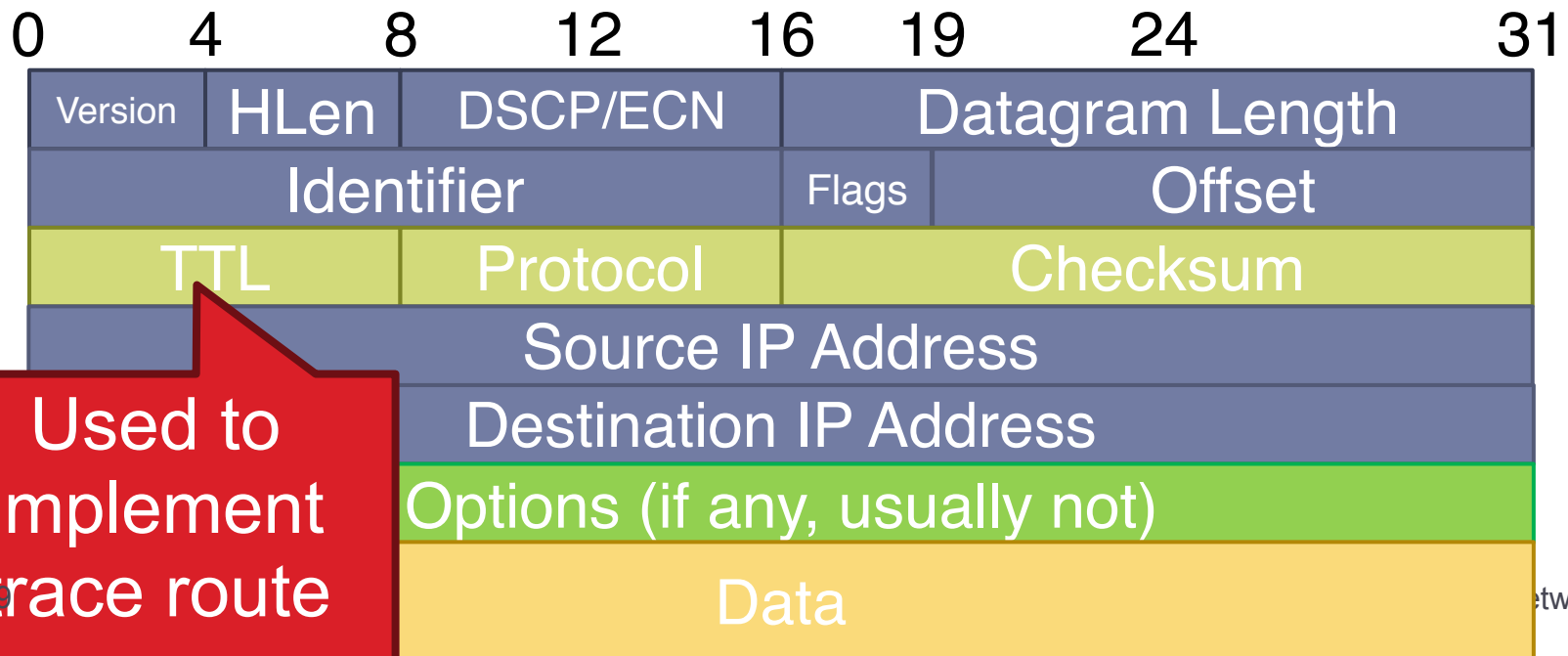
# IP Header Fields: Word 3

- ▶ Time to Live: decremented by each router
  - ▶ Used to kill looping packets
- ▶ Protocol: ID of encapsulated protocol
  - ▶ 6 = TCP, 17 = UDP
- ▶ Checksum



# IP Header Fields: Word 3

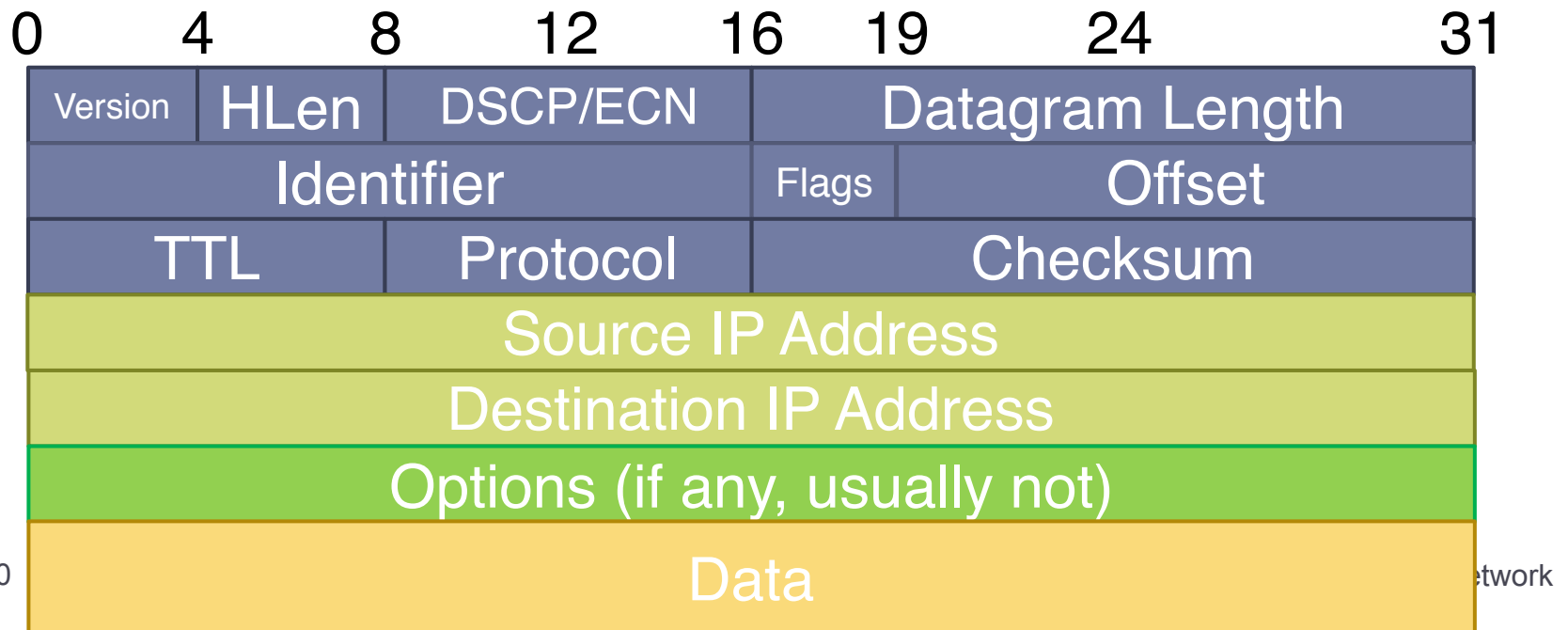
- ▶ Time to Live: decremented by each router
  - ▶ Used to kill looping packets
- ▶ Protocol: ID of encapsulated protocol
  - ▶ 6 = TCP, 17 = UDP
- ▶ Checksum



Used to implement trace route

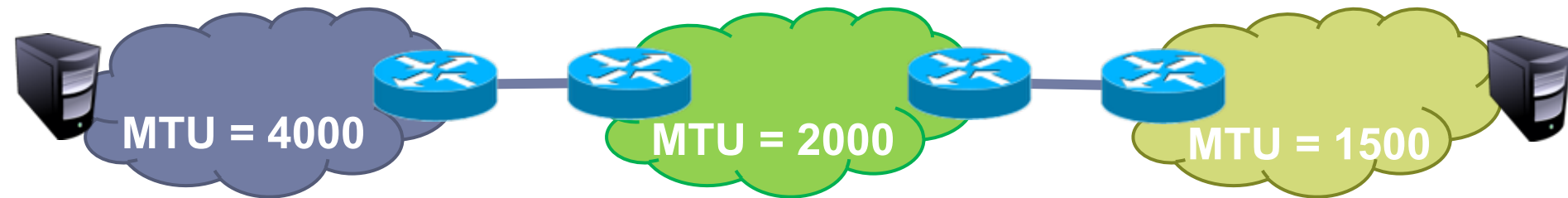
# IP Header Fields: Word 4 and 5

- ▶ Source and destination address
  - ▶ In theory, must be globally unique
  - ▶ In practice, this is often violated



# Problem: Fragmentation

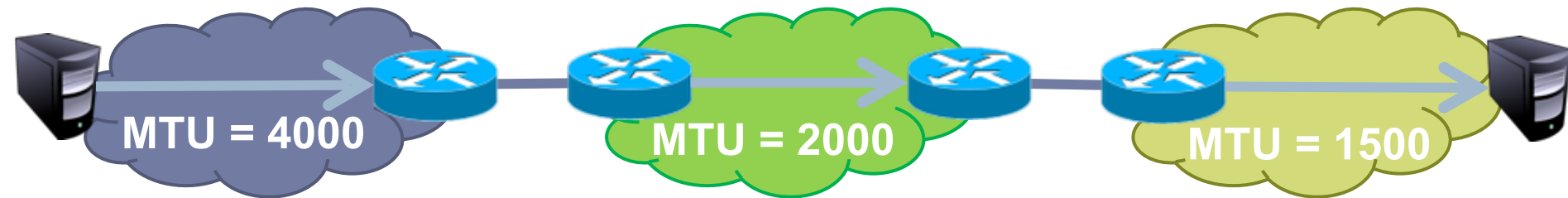
---



- ▶ **Problem: each network has its own MTU**
  - ▶ DARPA principles: networks allowed to be heterogeneous
  - ▶ Minimum MTU may not be known for a given path

# Problem: Fragmentation

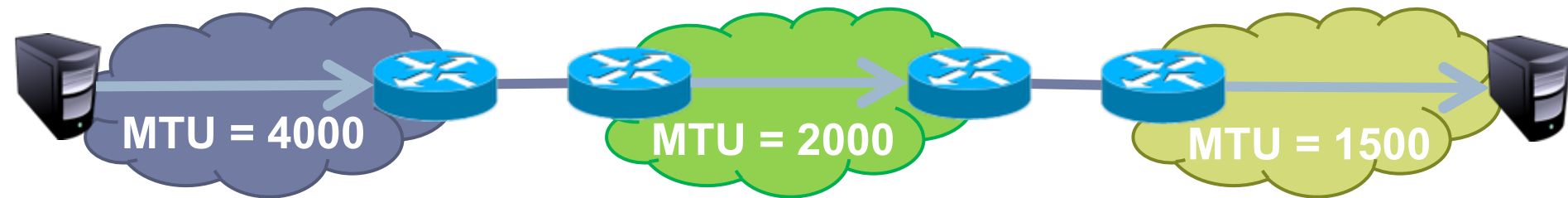
---



- ▶ **Problem: each network has its own MTU**
  - ▶ DARPA principles: networks allowed to be heterogeneous
  - ▶ Minimum MTU may not be known for a given path

# Problem: Fragmentation

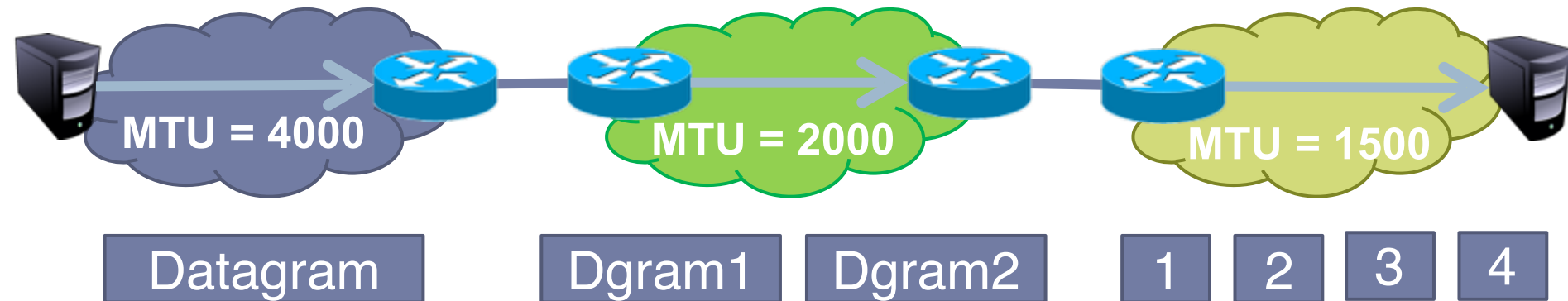
---



- ▶ **Problem: each network has its own MTU**
  - ▶ DARPA principles: networks allowed to be heterogeneous
  - ▶ Minimum MTU may not be known for a given path
- ▶ **IP Solution: fragmentation**
  - ▶ Split datagrams into pieces when MTU is reduced
  - ▶ Reassemble original datagram at the receiver

# Problem: Fragmentation

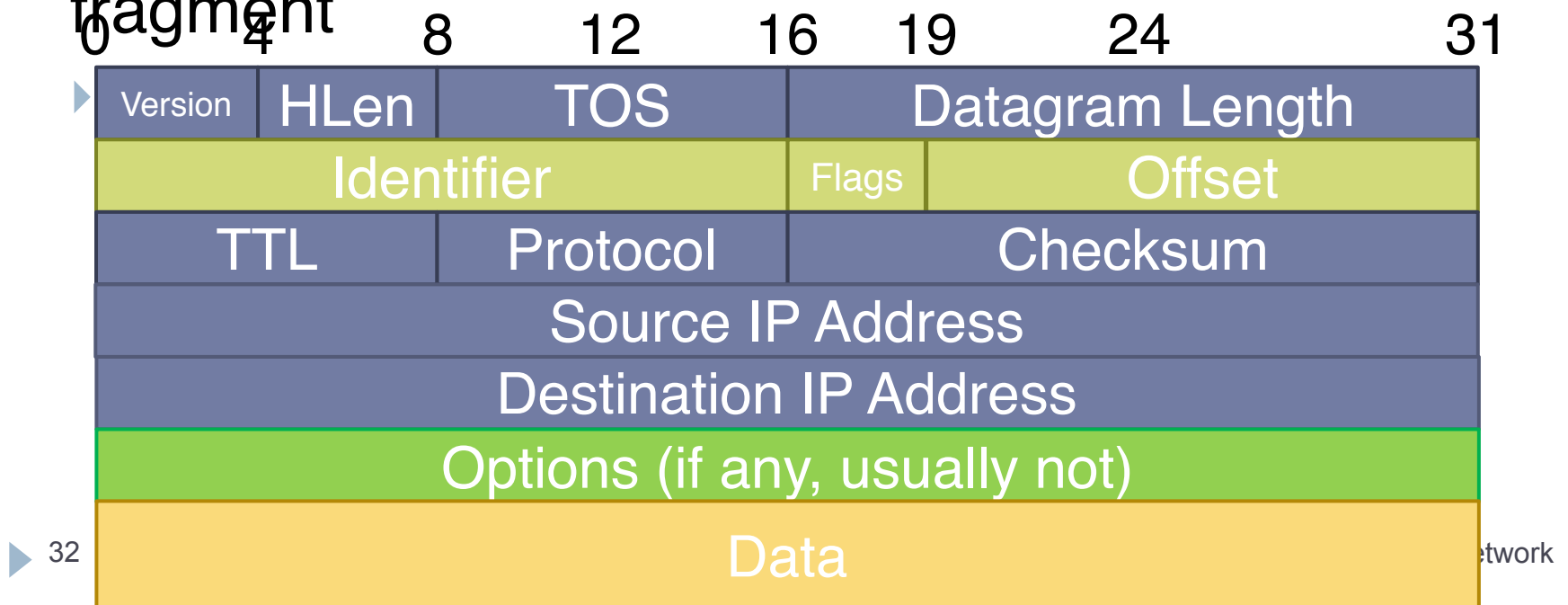
---



- ▶ **Problem: each network has its own MTU**
  - ▶ DARPA principles: networks allowed to be heterogeneous
  - ▶ Minimum MTU may not be known for a given path
- ▶ **IP Solution: fragmentation**
  - ▶ Split datagrams into pieces when MTU is reduced
  - ▶ Reassemble original datagram at the receiver

# IP Header Fields: Word 2

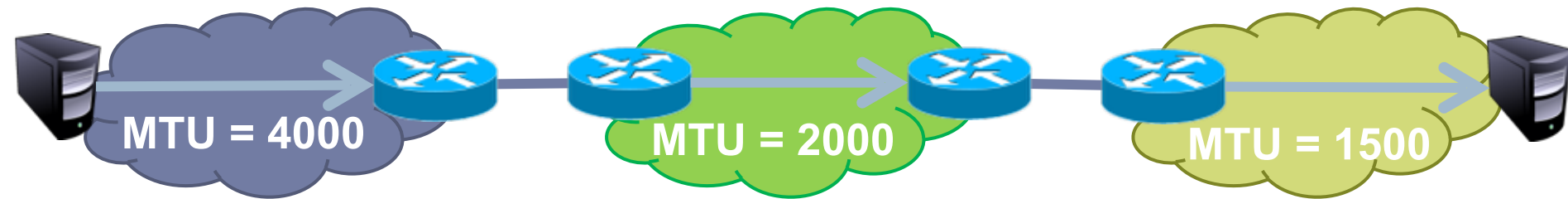
- ▶ Identifier: a unique number for the original datagram
- ▶ Flags: M flag, i.e. this is the last fragment
- ▶ Offset: byte position of the first byte in the fragment





# Fragmentation Example

---



Length = 3820, M = 0

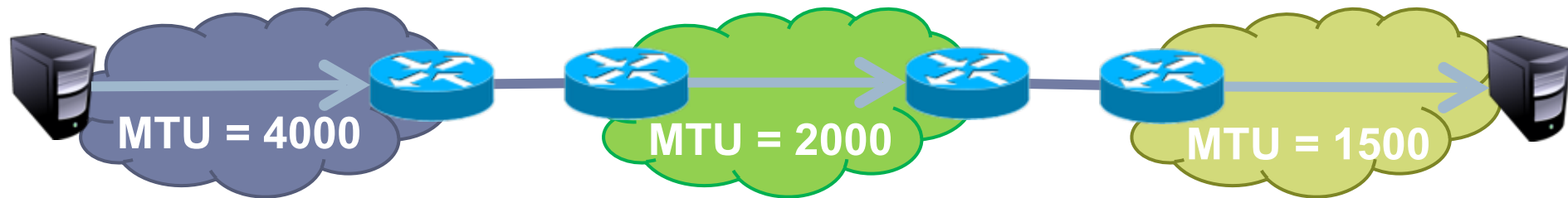
IP Hdr

Data

20

3800

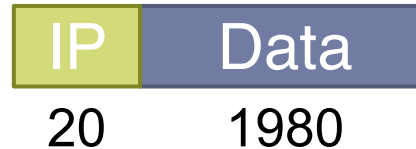
# Fragmentation Example



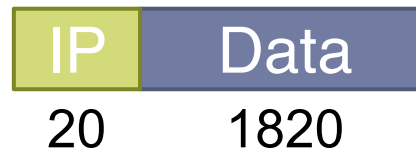
Length = 3820, M = 0



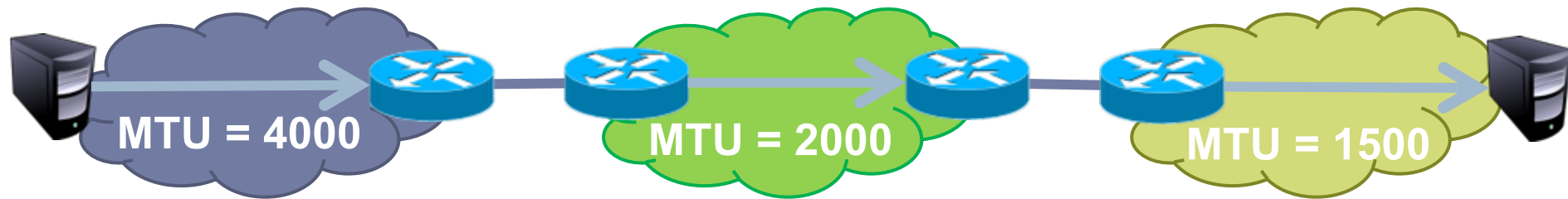
Length = 2000, M = 1  
Offset = 0



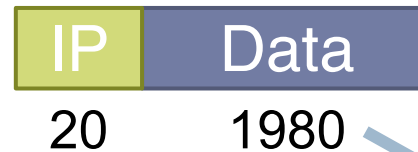
Length = 1840, M = 0  
Offset = 1980



# Fragmentation Example



Length = 2000, M = 1  
Offset = 0



Length = 3820, M = 0

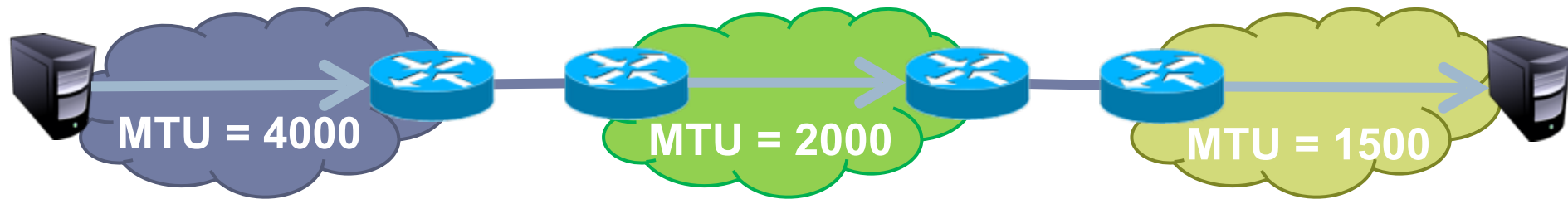


Length = 1840, M = 0  
Offset = 1980



1980  
+ 1820  
= 3800

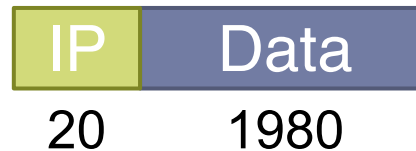
# Fragmentation Example



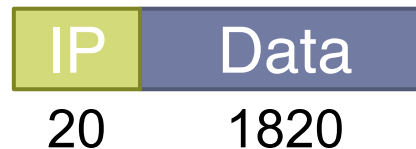
Length = 3820, M = 0



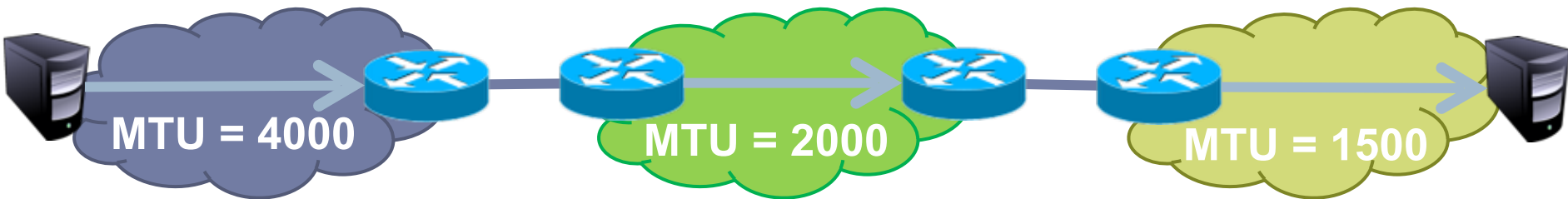
Length = 2000, M = 1  
Offset = 0



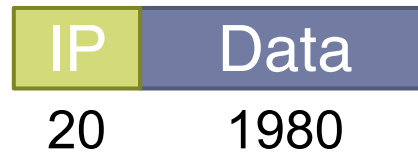
Length = 1840, M = 0  
Offset = 1980



# Fragmentation Example



Length = 2000, M = 1  
Offset = 0



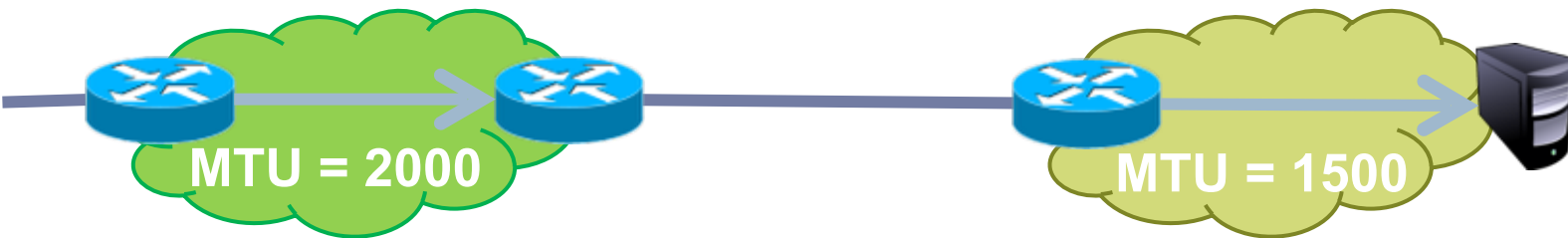
Length = 3820, M = 0



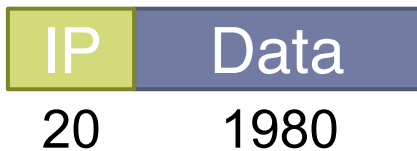
Length = 1840, M = 0  
Offset = 1980



# Fragmentation Example (cont.)



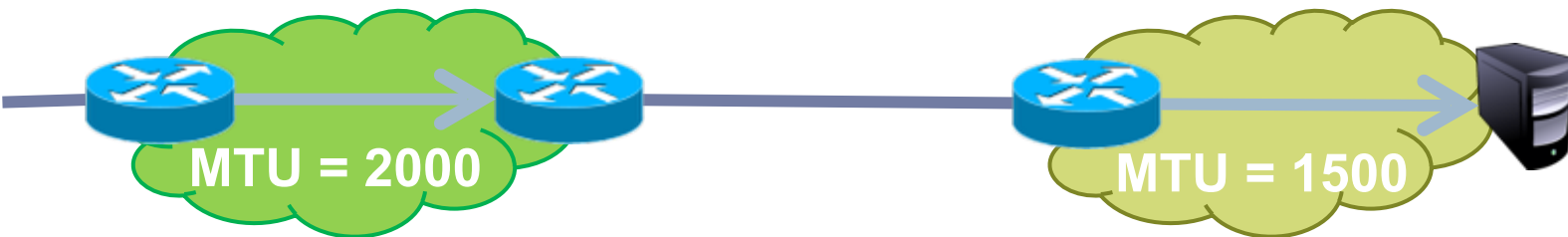
Length = 2000, M = 1  
Offset = 0



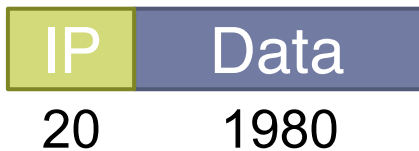
Length = 1840, M = 0  
Offset = 1980



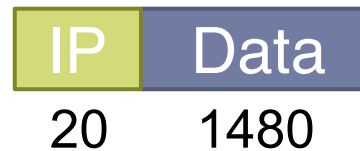
# Fragmentation Example (cont.)



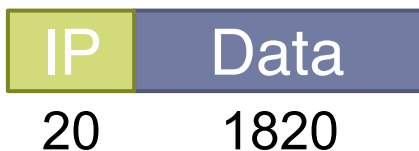
Length = 2000, M = 1  
Offset = 0



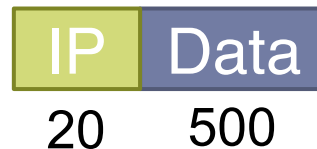
Length = 1500, M = 1  
Offset = 0



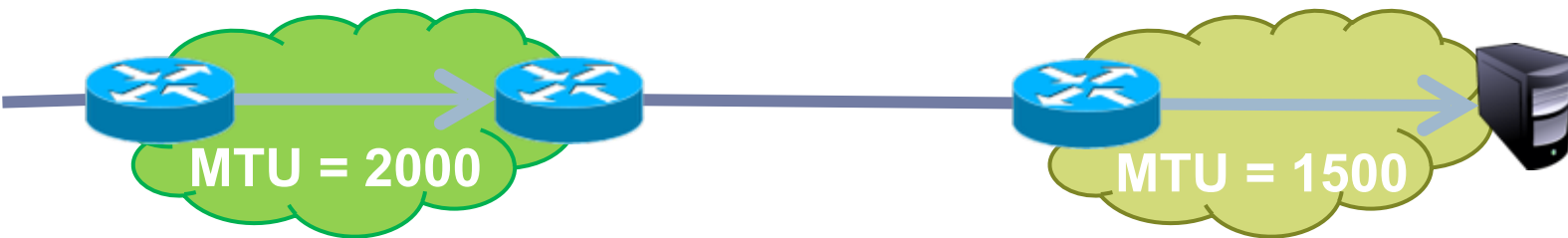
Length = 1840, M = 0  
Offset = 1980



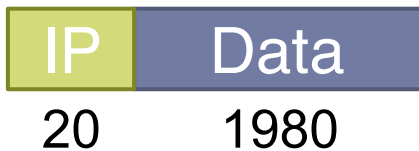
Length = 520, M = 1  
Offset = 1480



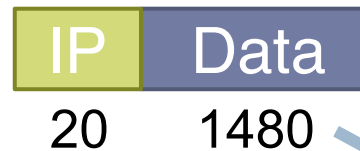
# Fragmentation Example (cont.)



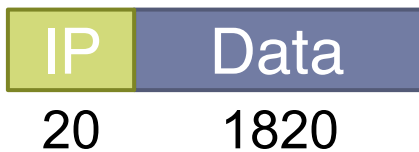
Length = 2000, M = 1  
Offset = 0



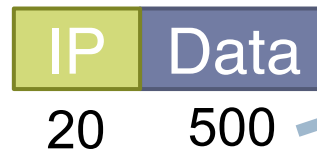
Length = 1500, M = 1  
Offset = 0



Length = 1840, M = 0  
Offset = 1980



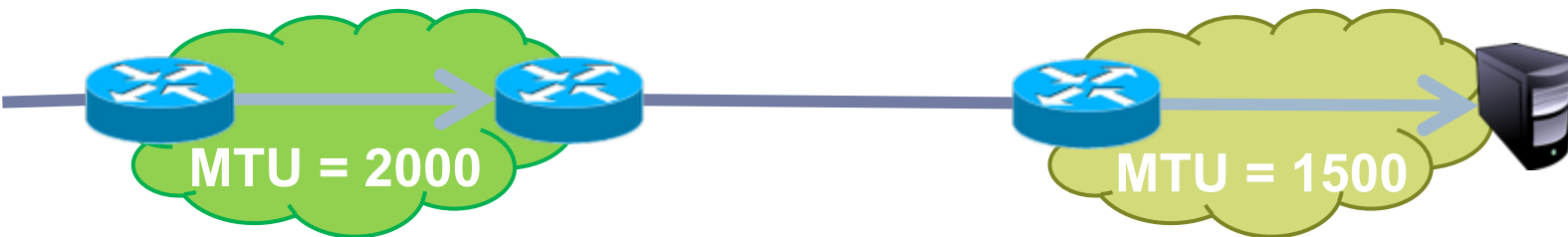
Length = 520, M = 1  
Offset = 1480



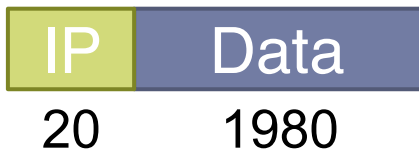
$$\begin{aligned} &1480 \\ &+ 500 \\ &= 1980 \end{aligned}$$



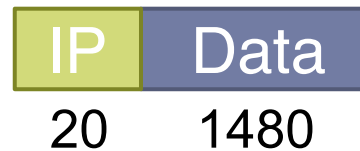
# Fragmentation Example (cont.)



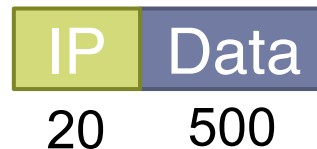
Length = 2000, M = 1  
Offset = 0



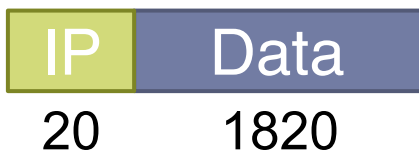
Length = 1500, M = 1  
Offset = 0



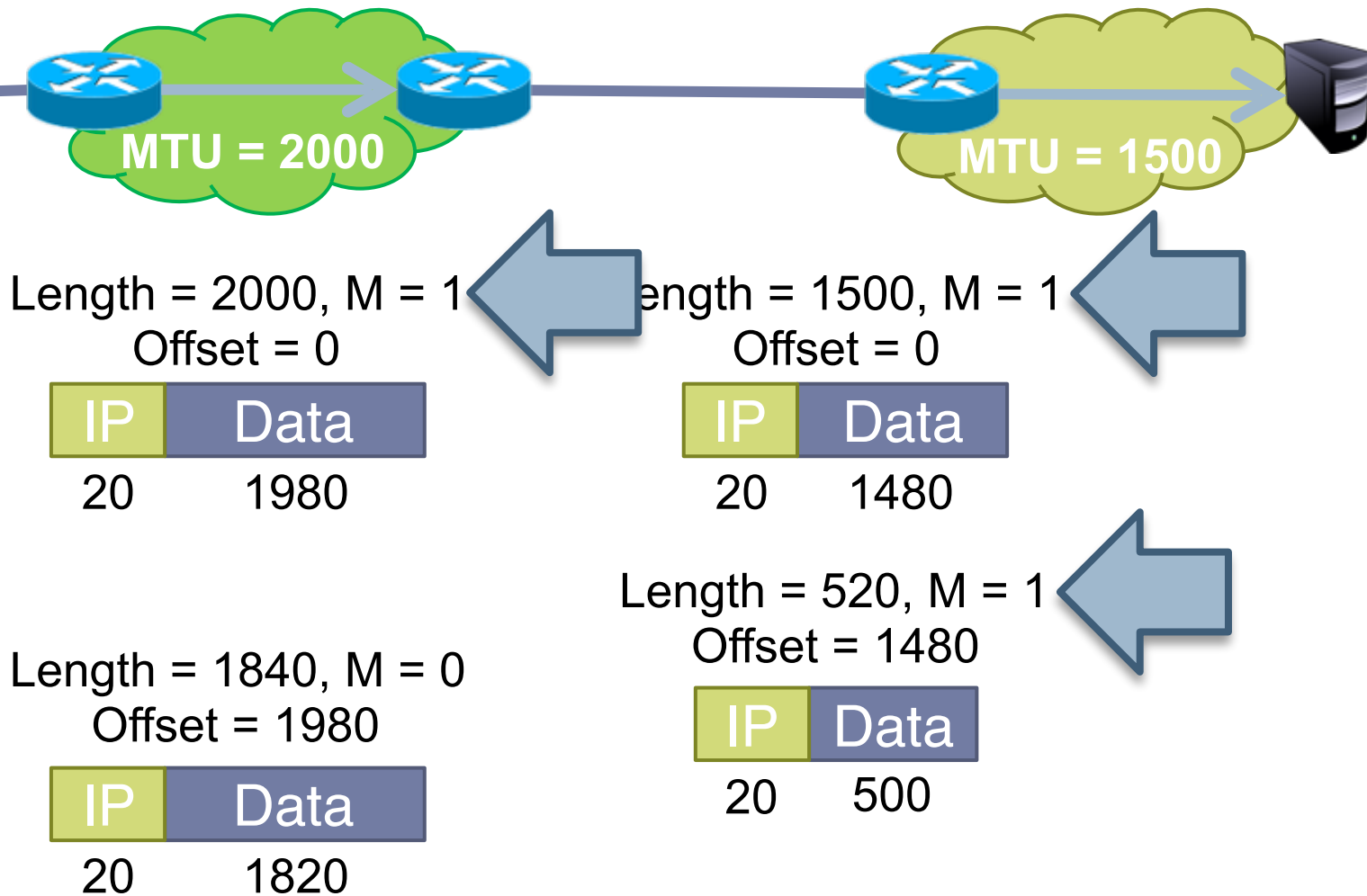
Length = 520, M = 1  
Offset = 1480



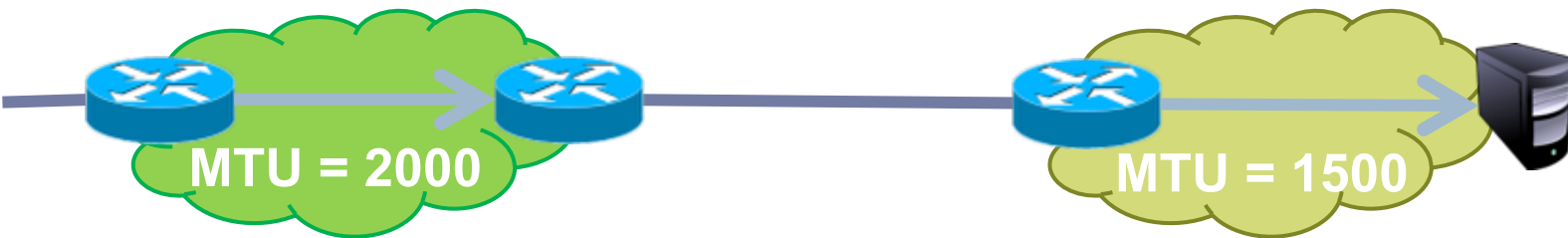
Length = 1840, M = 0  
Offset = 1980



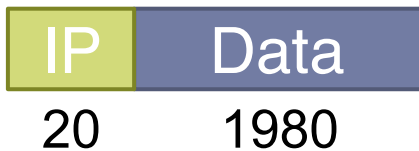
# Fragmentation Example (cont.)



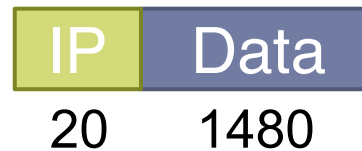
# Fragmentation Example (cont.)



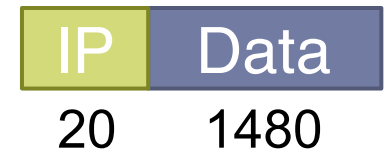
Length = 2000, M = 1  
Offset = 0



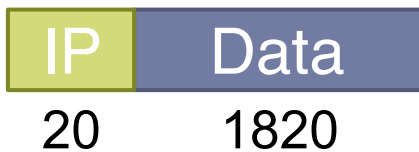
Length = 1500, M = 1  
Offset = 0



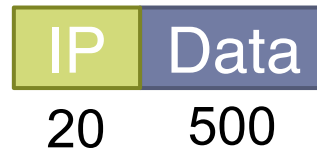
Length = 1500, M = 1  
Offset = 1980



Length = 1840, M = 0  
Offset = 1980



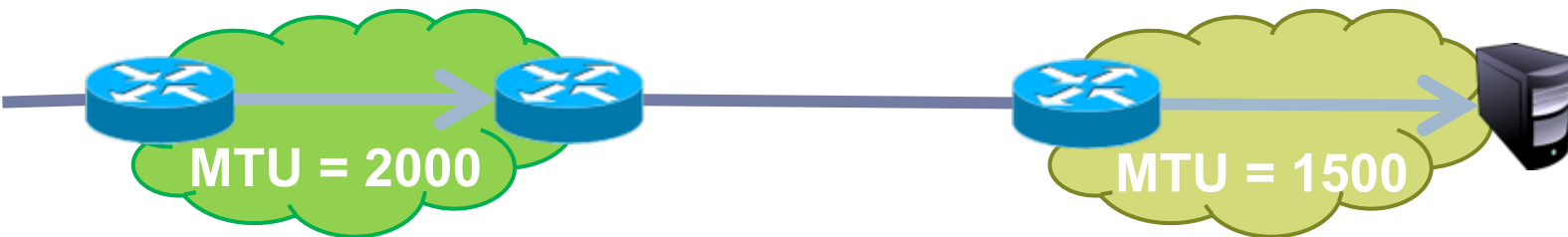
Length = 520, M = 1  
Offset = 1480



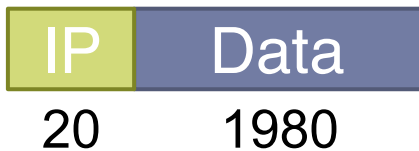
Length = 360, M = 0  
Offset = 3460



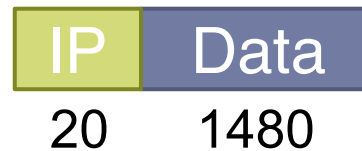
# Fragmentation Example (cont.)



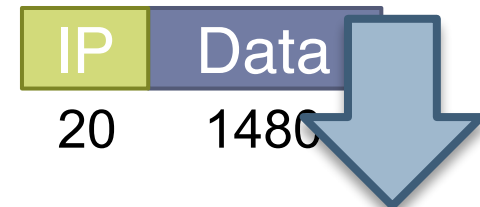
Length = 2000, M = 1  
Offset = 0



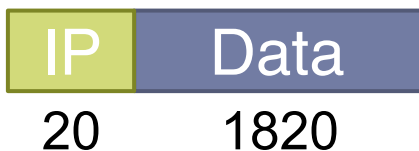
Length = 1500, M = 1  
Offset = 0



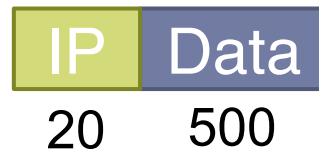
Length = 1500, M = 1  
Offset = 1980



Length = 1840, M = 0  
Offset = 1980



Length = 520, M = 1  
Offset = 1480



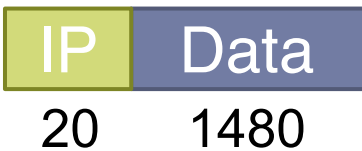
Length = 360, M = 0  
Offset = 3460



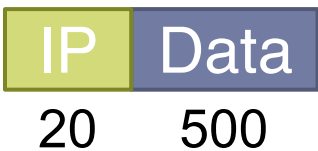
# IP Fragment Reassembly

---

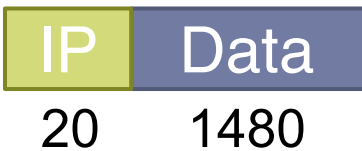
Length = 1500, M = 1, Offset = 0



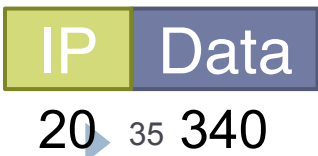
Length = 520, M = 1, Offset = 1480



Length = 1500, M = 1, Offset = 1980



Length = 360, M = 0, Offset = 3460



- ▶ Performed at destination
- ▶ M = 0 fragment gives us total data size
  - ▶  $360 - 20 + 3460 = 3800$
- ▶ Challenges:
  - ▶ Out-of-order fragments
  - ▶ Duplicate fragments
  - ▶ Missing fragments
- ▶ Basically, memory management nightmare

# Fragmentation Concepts

---

- ▶ Highlights many key Internet characteristics
  - ▶ Decentralized and heterogeneous
    - ▶ Each network may choose its own MTU
  - ▶ Connectionless datagram protocol
    - ▶ Each fragment contains full routing information
    - ▶ Fragments can travel independently, on different paths
  - ▶ Best effort network
    - ▶ Routers/receiver may silently drop fragments
    - ▶ No requirement to alert the sender
  - ▶ Most work is done at the endpoints
    - ▶ i.e. reassembly

# Fragmentation in Reality

---

- ▶ **Fragmentation is expensive**
  - ▶ Memory and CPU overhead for datagram reconstruction
  - ▶ Want to avoid fragmentation if possible
- ▶ **MTU discovery protocol**
  - ▶ Send a packet with “don’t fragment” bit set
  - ▶ Keep decreasing message length until one arrives
  - ▶ May get “can’t fragment” error from a router, which will explicitly state the supported MTU
- ▶ **Router handling of fragments**
  - ▶ Fast, specialized hardware handles the common case
  - ▶ Dedicated, general purpose CPU just for handling fragments



## 3: IPv6



# The IPv4 Address Space Crisis

---

- ▶ **Problem: the IPv4 address space is too small**
  - ▶  $2^{32} = 4,294,967,296$  possible addresses
  - ▶ Less than one IP per person
- ▶ **Parts of the world have already run out of addresses**
  - ▶ IANA assigned the last /8 block of addresses in 2011

Region	Regional Internet Registry (RIR)	Exhaustion Date
Asia/Pacific	APNIC	April 19, 2011
Europe/Middle East	RIPE	September 14, 2012
North America	ARIN	13 Jan 2015 (Projected)
South America	LACNIC	13 Jan 2015 (Projected)
Africa	AFRINIC	17 Jan 2022(Projected)

# IPv6

---

- ▶ IPv6, first introduced in 1998(!)
  - ▶ 128-bit addresses
  - ▶  $4.8 * 10^{28}$  addresses per person
- ▶ Address format
  - ▶ 8 groups of 16-bit values, separated by ‘:’
  - ▶ Leading zeroes in each group may be omitted
  - ▶ Groups of zeroes can be omitted using ‘::’

2001:0db8:0000:0000:0000:ff00:0042:8329

2001:0db8:0:0:0:ff00:42:8329

2001:0db8::ff00:42:8329

# IPv6 Trivia

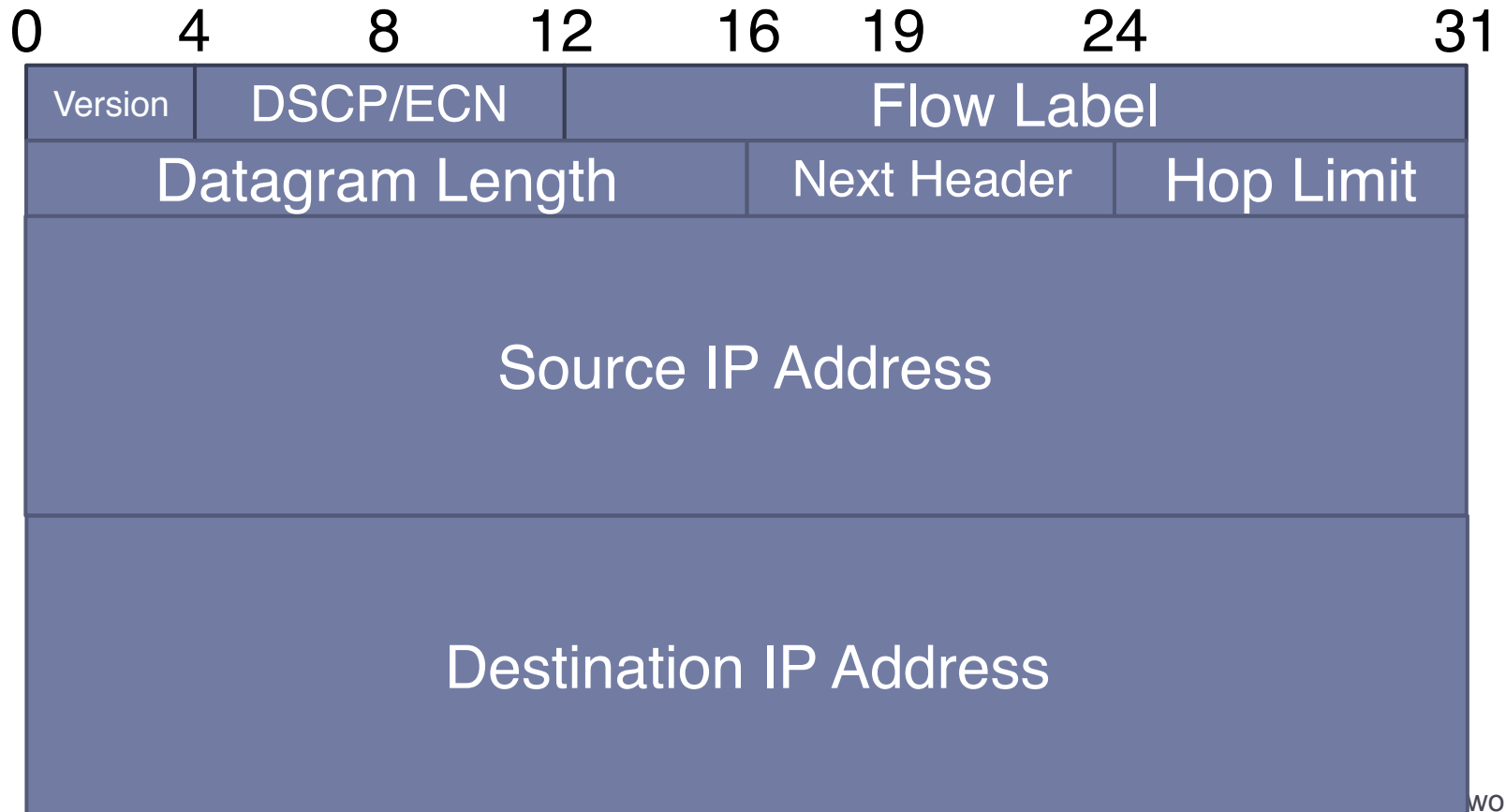
---

- ▶ Who knows the IP for localhost?
  - ▶ 127.0.0.1
- ▶ What is localhost in IPv6?
  - ▶ ::1

# IPv6 Header

---

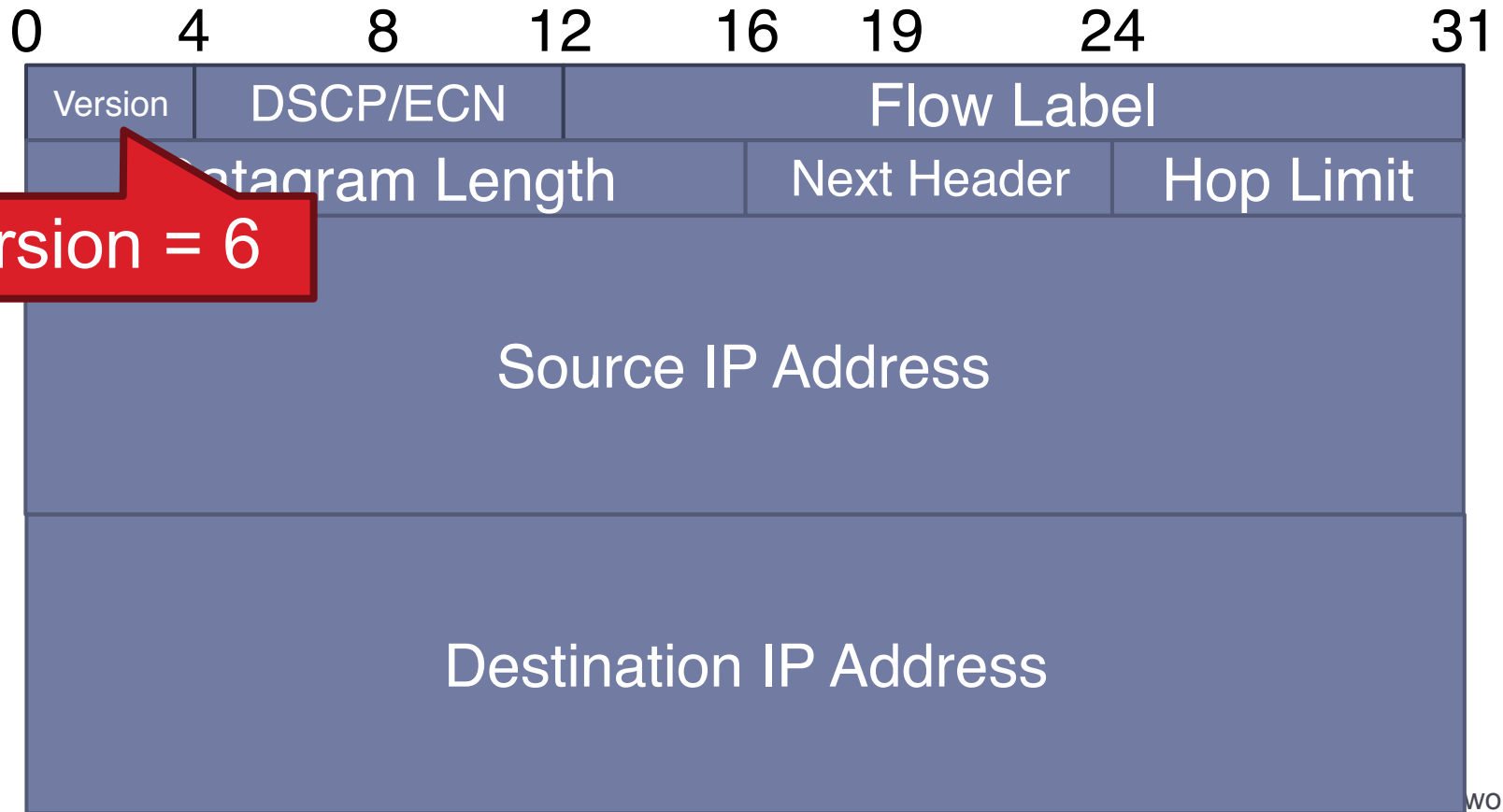
- ▶ Double the size of IPv4 (320 bits vs. 160 bits)



# IPv6 Header

---

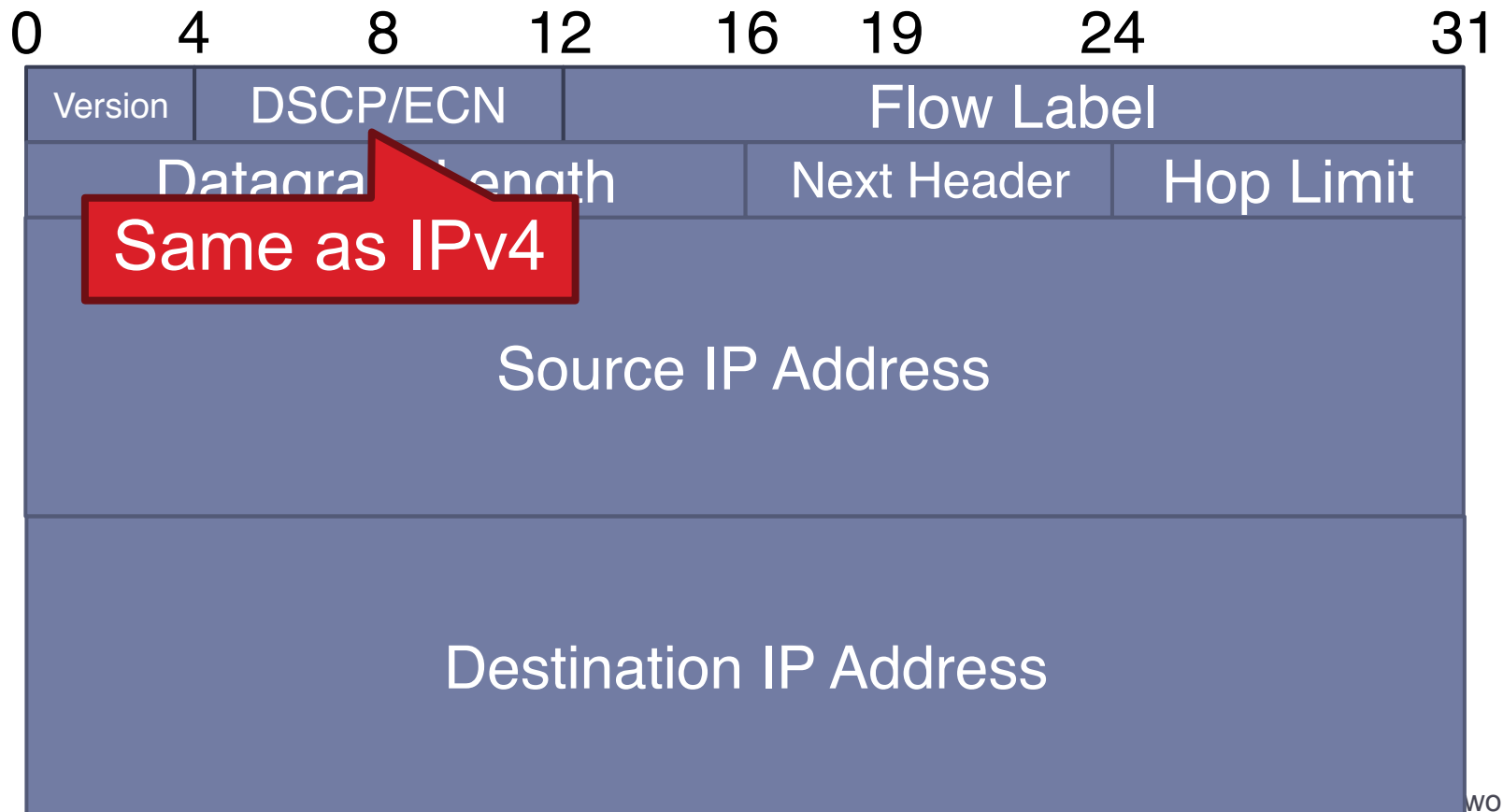
- ▶ Double the size of IPv4 (320 bits vs. 160 bits)



# IPv6 Header

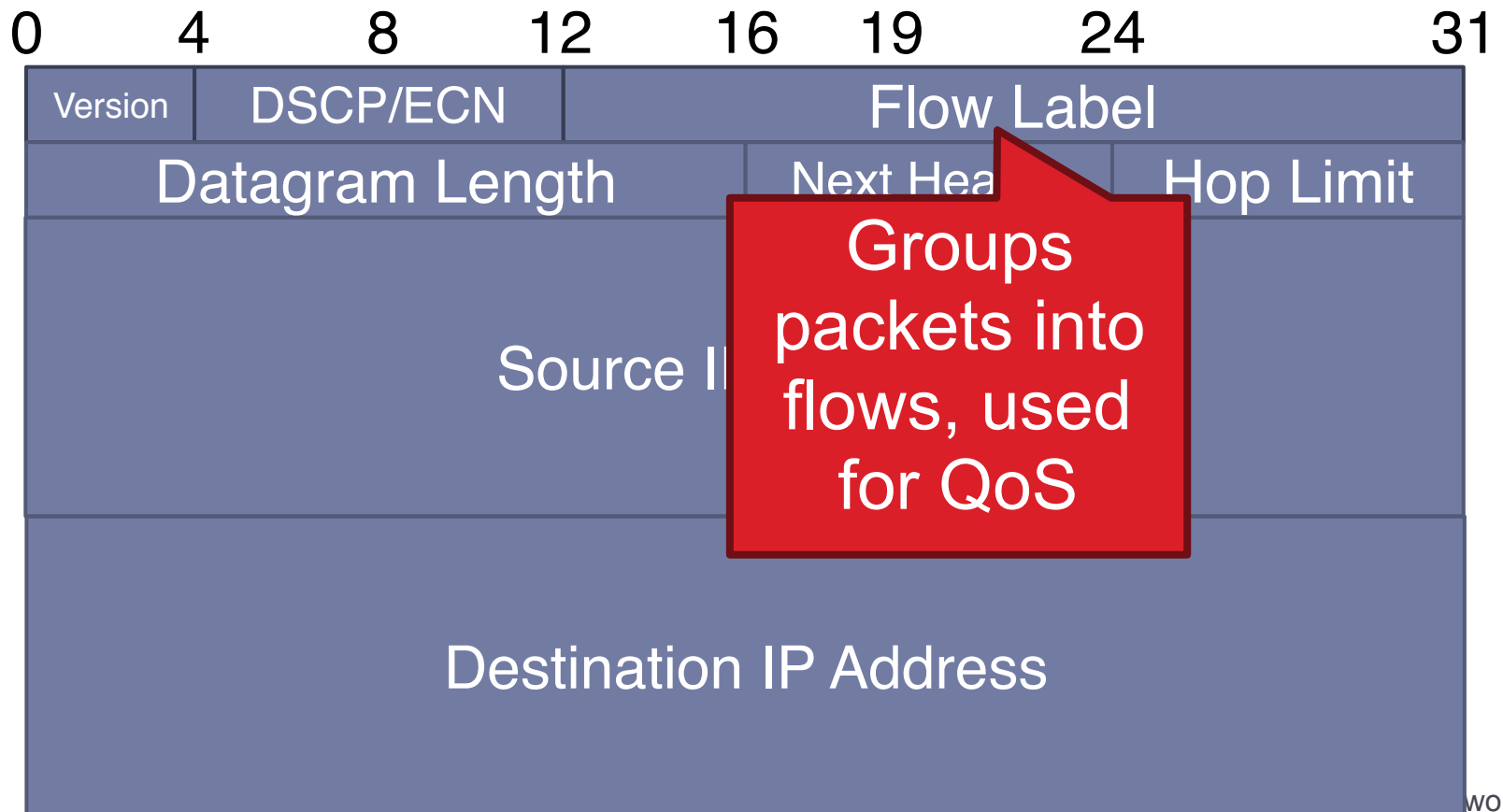
---

- ▶ Double the size of IPv4 (320 bits vs. 160 bits)



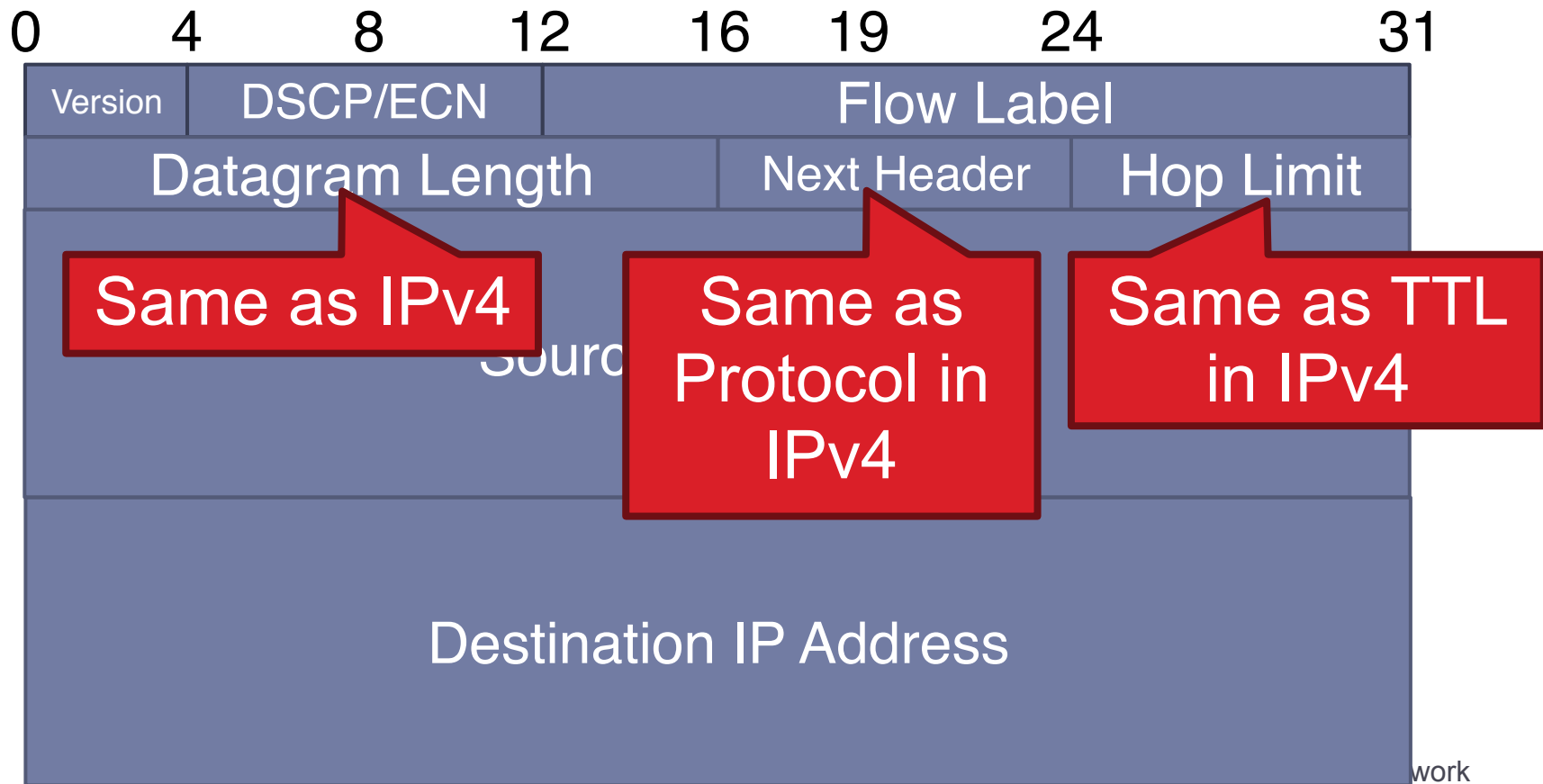
# IPv6 Header

- ▶ Double the size of IPv4 (320 bits vs. 160 bits)



# IPv6 Header

- ▶ Double the size of IPv4 (320 bits vs. 160 bits)





# Differences from IPv4 Header

---

- ▶ **Several header fields are missing in IPv6**
  - ▶ Header length – rolled into Next Header field
  - ▶ Checksum – was useless, so why keep it
  - ▶ Identifier, Flags, Offset
    - ▶ IPv6 routers do not support fragmentation
    - ▶ Hosts are expected to use path MTU discovery
- ▶ **Reflects changing Internet priorities**
  - ▶ Today's networks are more homogeneous
  - ▶ Instead, routing cost and complexity dominate
- ▶ **No security vulnerabilities due to IP fragments**

# Performance Improvements

---

- ▶ No checksums to verify
- ▶ No need for routers to handle fragmentation
- ▶ Simplified routing table design
  - ▶ Address space is huge
  - ▶ No need for CIDR (but need for aggregation)
  - ▶ Standard subnet size is  $2^{64}$  addresses
- ▶ Simplified auto-configuration
  - ▶ Neighbor Discovery Protocol
  - ▶ Used by hosts to determine network ID
  - ▶ Host ID can be random!

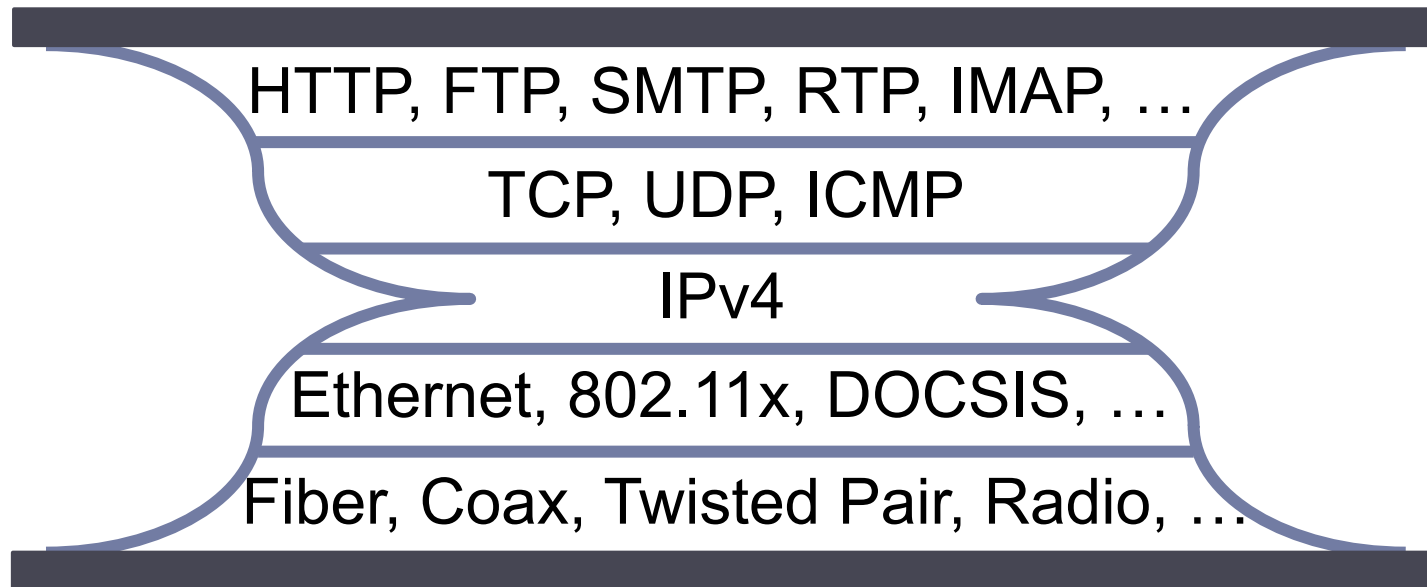
# Additional IPv6 Features

---

- ▶ **Source Routing**
  - ▶ Host specifies the route to wants packet to take
- ▶ **Mobile IP**
  - ▶ Hosts can take their IP with them to other networks
  - ▶ Use source routing to direct packets
- ▶ **Privacy Extensions**
  - ▶ Randomly generate host identifiers
  - ▶ Make it difficult to associate one IP to a host
- ▶ **Jumbograms**
  - ▶ Support for 4Gb datagrams

# Deployment Challenges

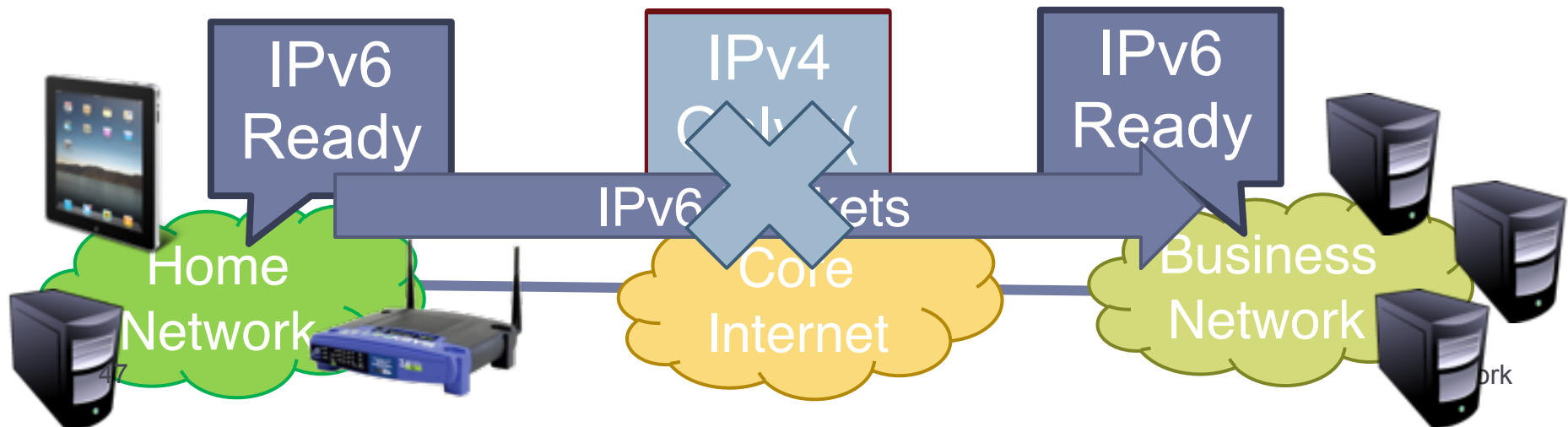
---



- ▶ Switching to IPv6 is a whole-Internet upgrade
  - ▶ All routers, all hosts
  - ▶ ICMPv6, DHCPv6, DNSv6
- ▶ 2013: 0.94% of Google traffic was IPv6, **13% today**

# Transitioning to IPv6

- ▶ How do we ease the transition from IPv4 to IPv6?
  - ▶ Today, most network edges are IPv6 ready
    - ▶ Windows/OSX/iOS/Android all support IPv6
    - ▶ Your wireless access point probably supports IPv6
  - ▶ The Internet core is hard to upgrade
  - ▶ ... but a IPv4 core cannot route IPv6 traffic



# Transition Technologies

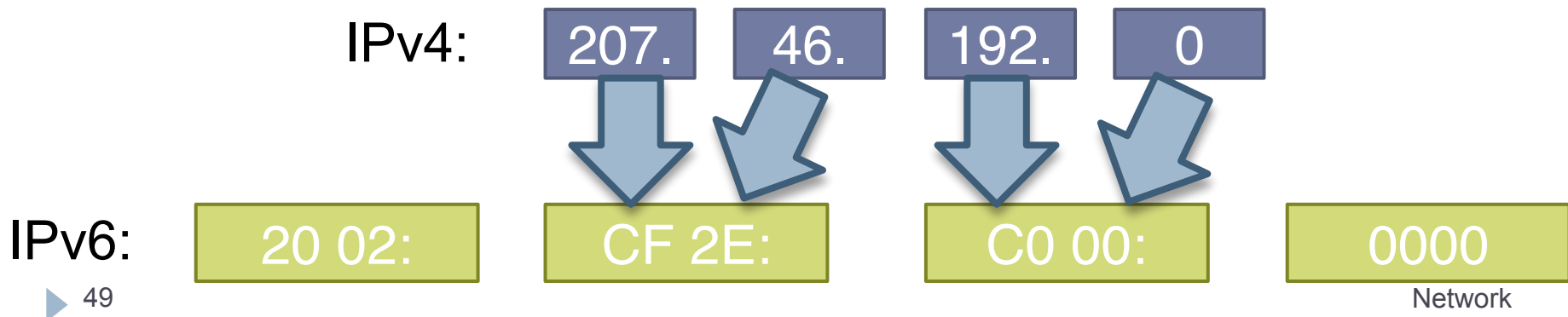
---

- ▶ How do you route IPv6 packets over an IPv4 Internet?
- ▶ Transition Technologies
  - ▶ Use tunnels to encapsulate and route IPv6 packets over the IPv4 Internet
  - ▶ Several different implementations
    - ▶ 6to4
    - ▶ IPv6 Rapid Deployment (6rd)
    - ▶ Teredo
    - ▶ ... etc.

# 6to4 Basics

---

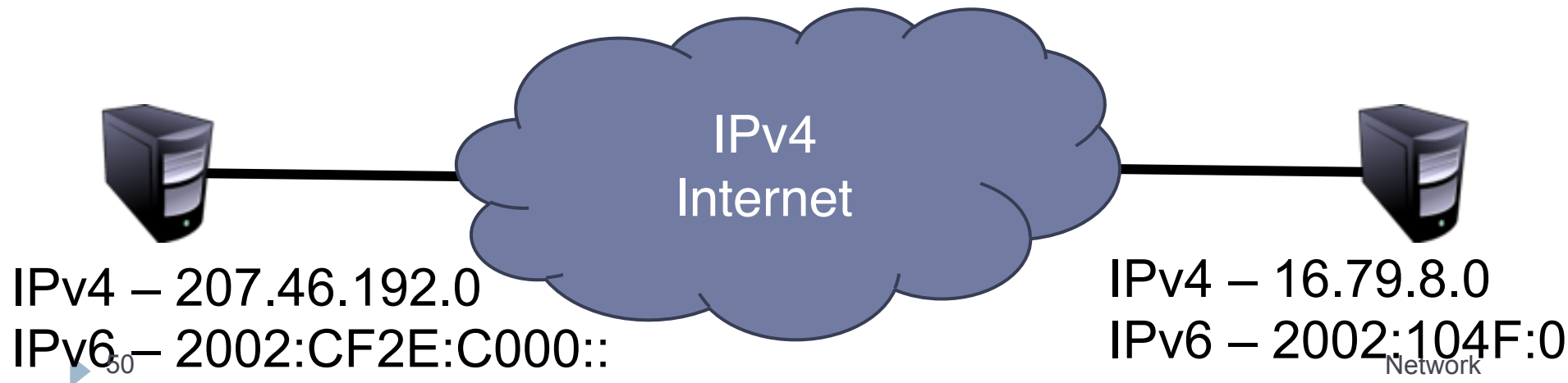
- ▶ **Problem:** you've been assigned an IPv4 address, but you want an IPv6 address
  - ▶ Your ISP can't or won't give you an IPv6 address
  - ▶ You can't just arbitrarily choose an IPv6 address
- ▶ **Solution:** construct a 6to4 address
  - ▶ 6to4 addresses always start with 2002::
  - ▶ Embed the 32-bit IPv4 inside the 128-bit IPv6 address



# Routing from 6to4 to 6to4

---

- ▶ How does a host using 6to4 send a packet to another host using 6to4?



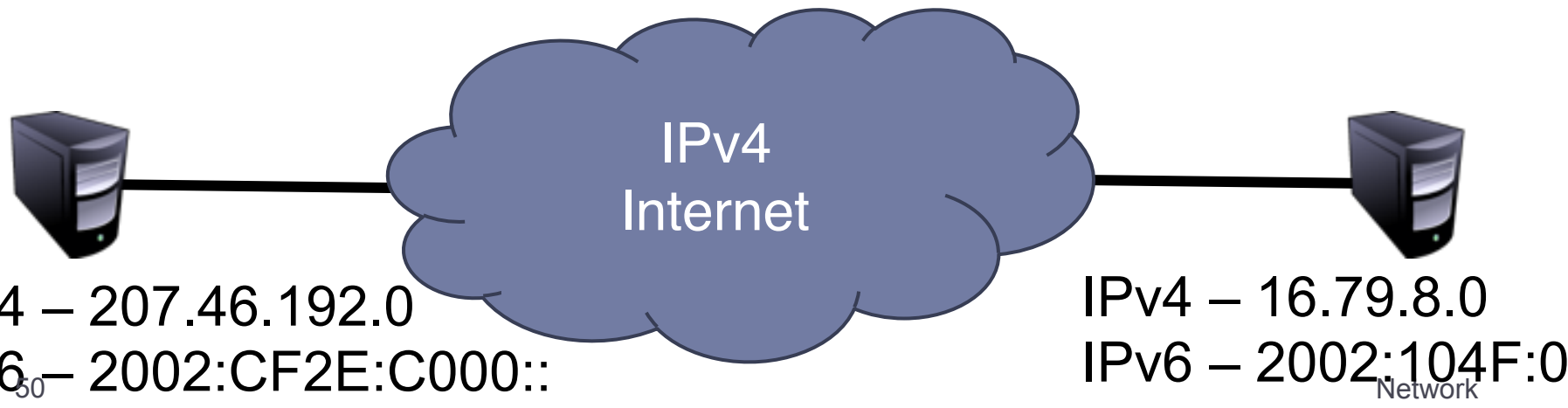


# Routing from 6to4 to 6to4

---

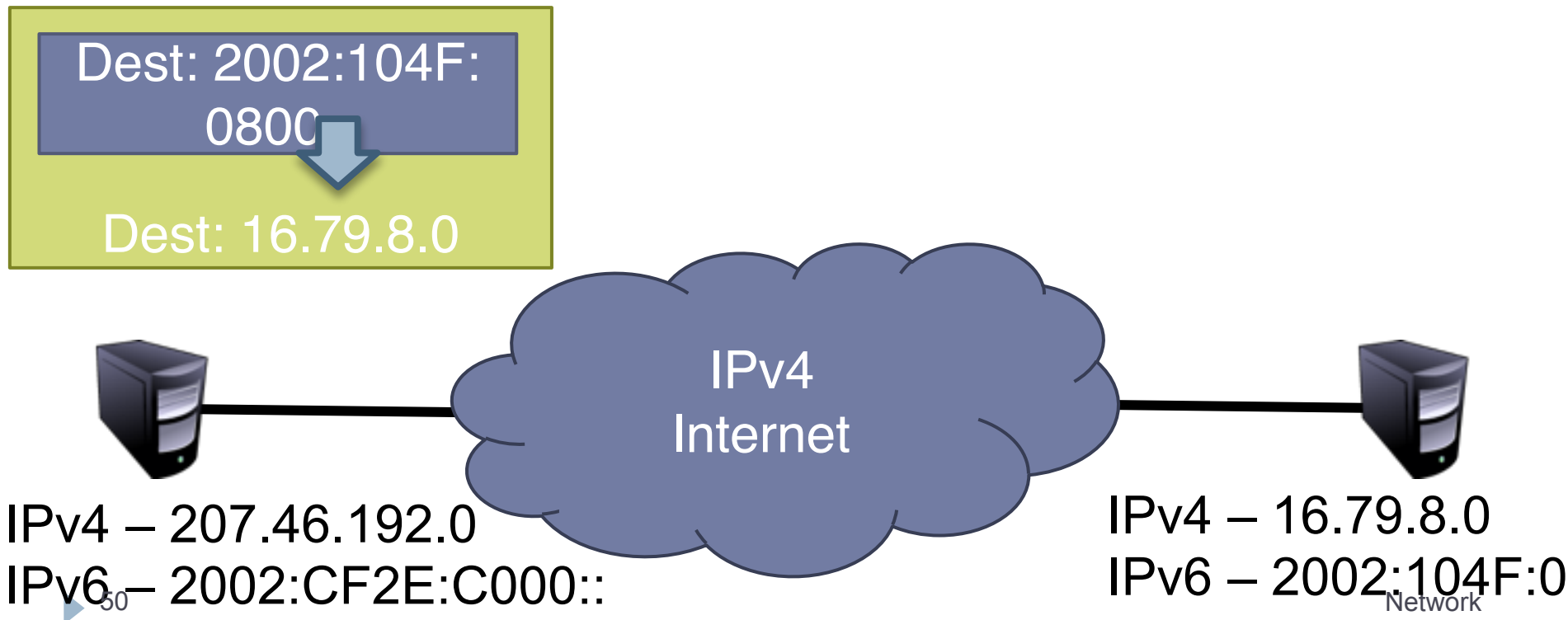
- ▶ How does a host using 6to4 send a packet to another host using 6to4?

Dest: 2002:104F:  
0800::



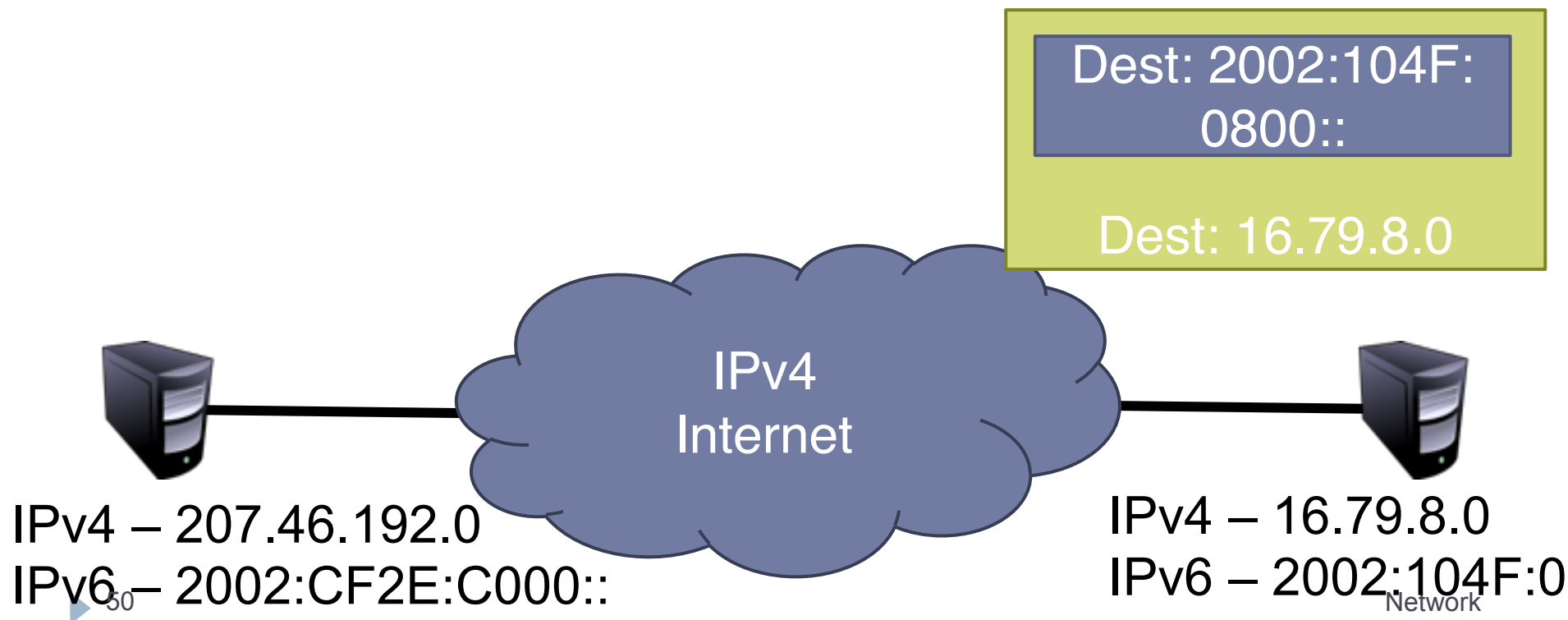
# Routing from 6to4 to 6to4

- ▶ How does a host using 6to4 send a packet to another host using 6to4?



# Routing from 6to4 to 6to4

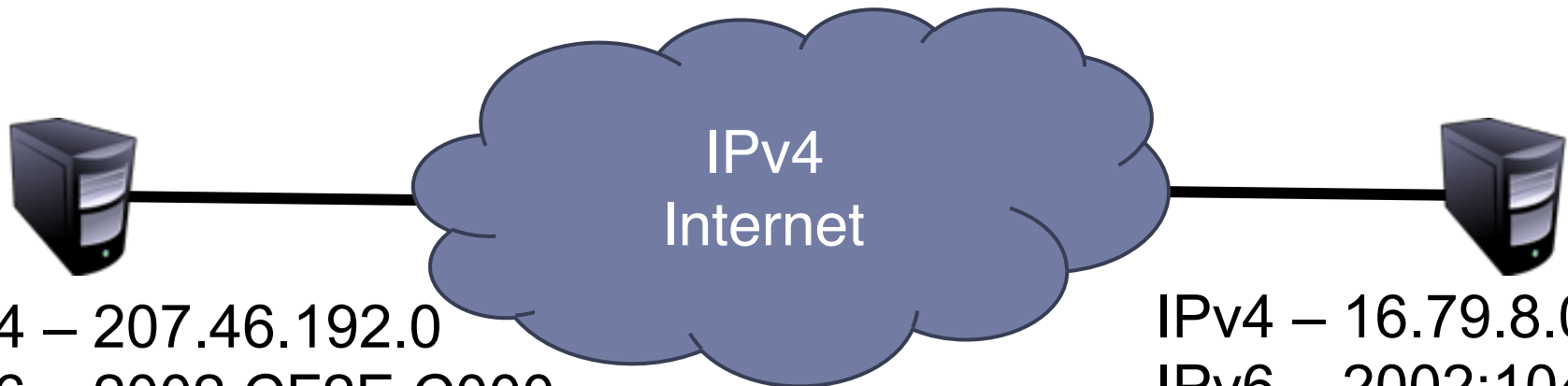
- ▶ How does a host using 6to4 send a packet to another host using 6to4?



# Routing from 6to4 to 6to4

- ▶ How does a host using 6to4 send a packet to another host using 6to4?

Dest: 2002:104F:  
0800::

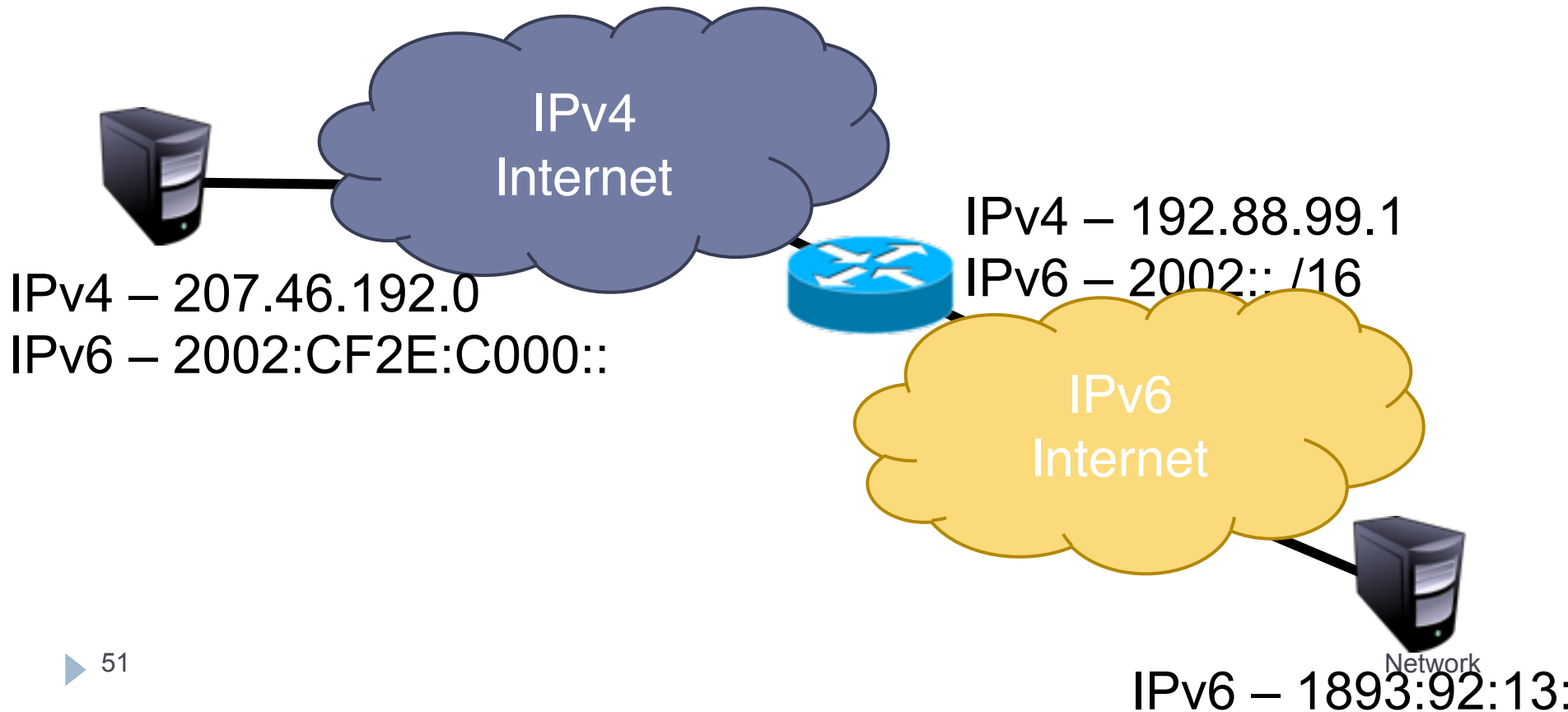


IPv4 – 207.46.192.0  
IPv6 – 2002:CF2E:C000::

IPv4 – 16.79.8.0  
IPv6 – 2002:104F:0800::  
Network

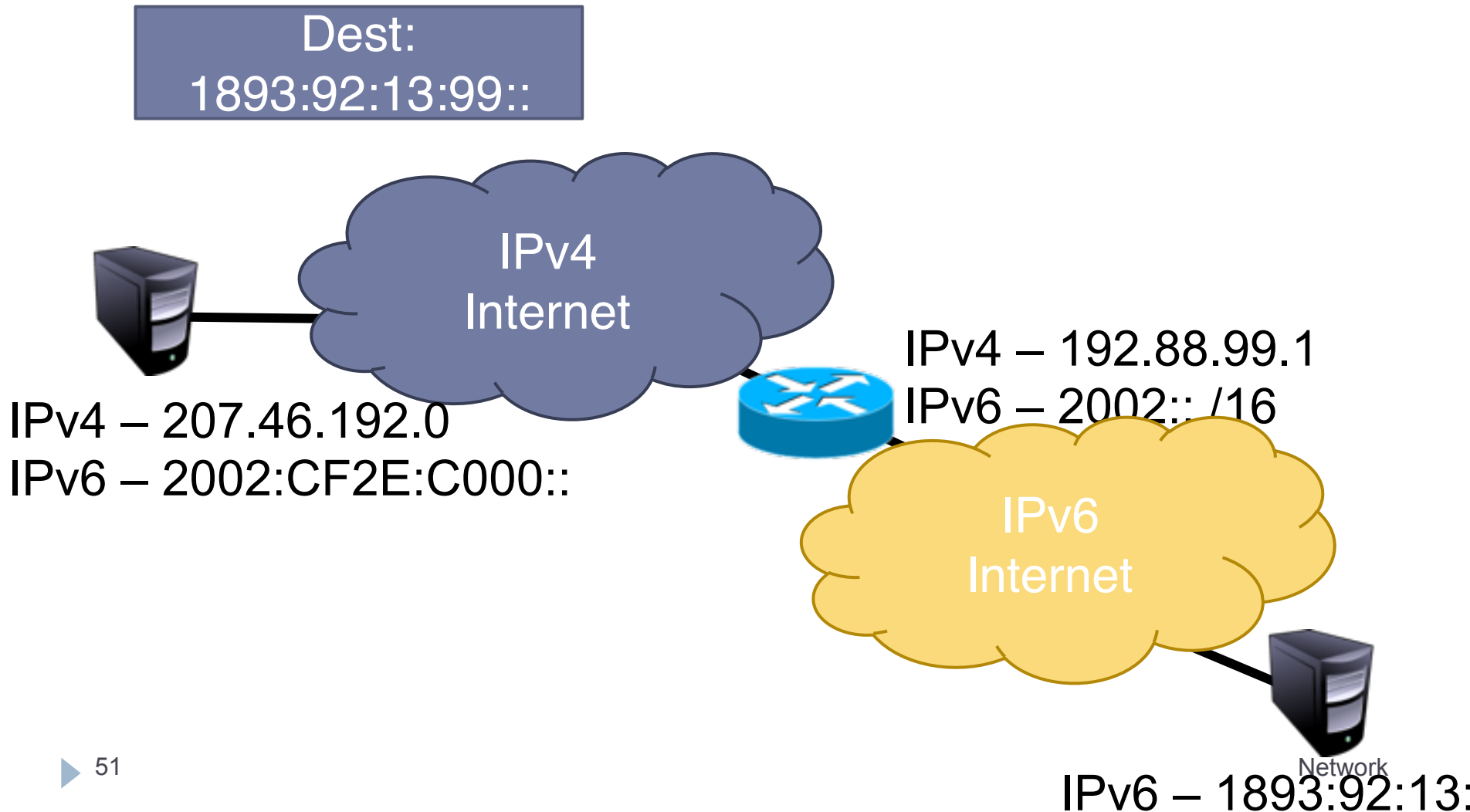
# Routing from 6to4 to Native IPv6

---

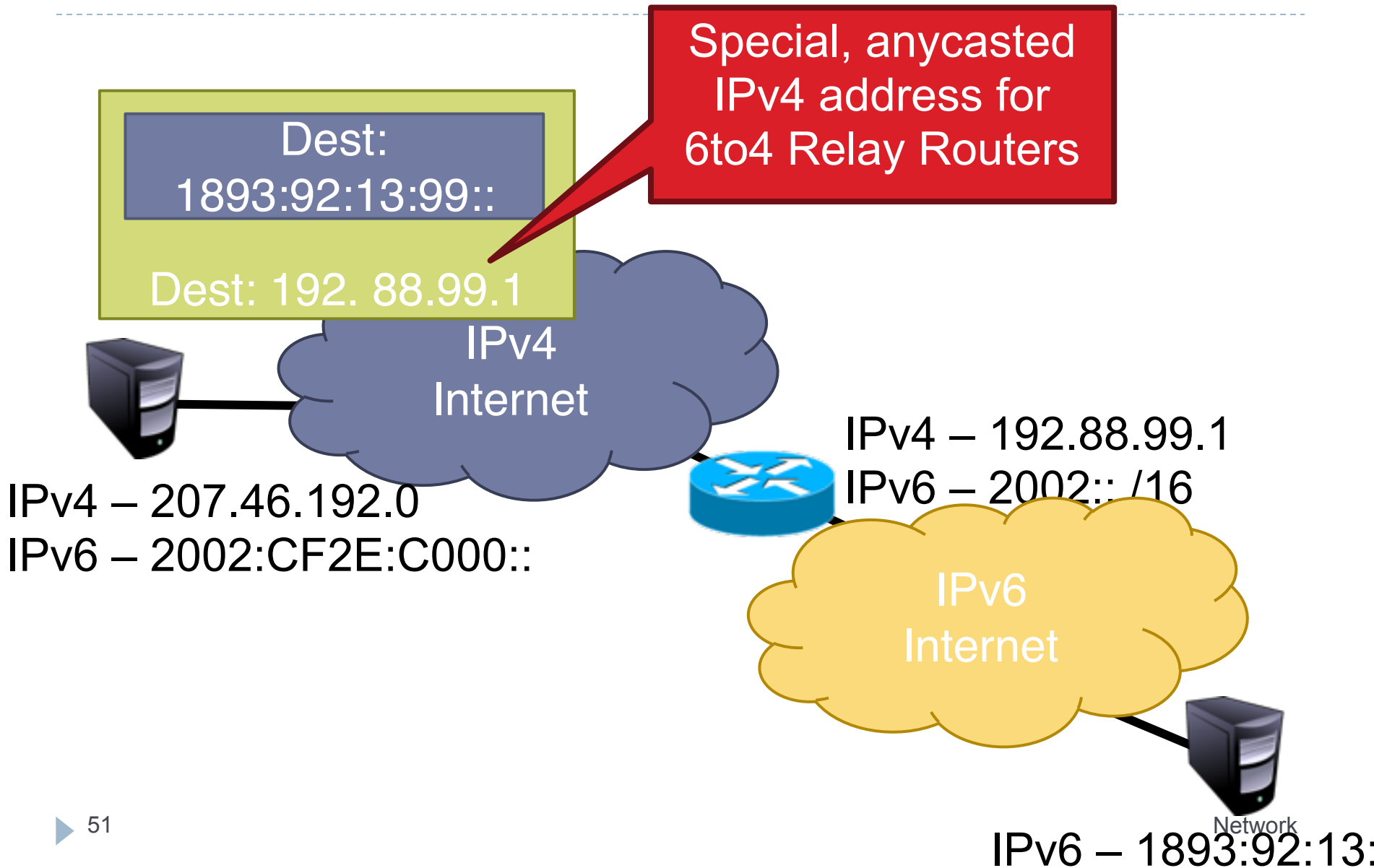


# Routing from 6to4 to Native IPv6

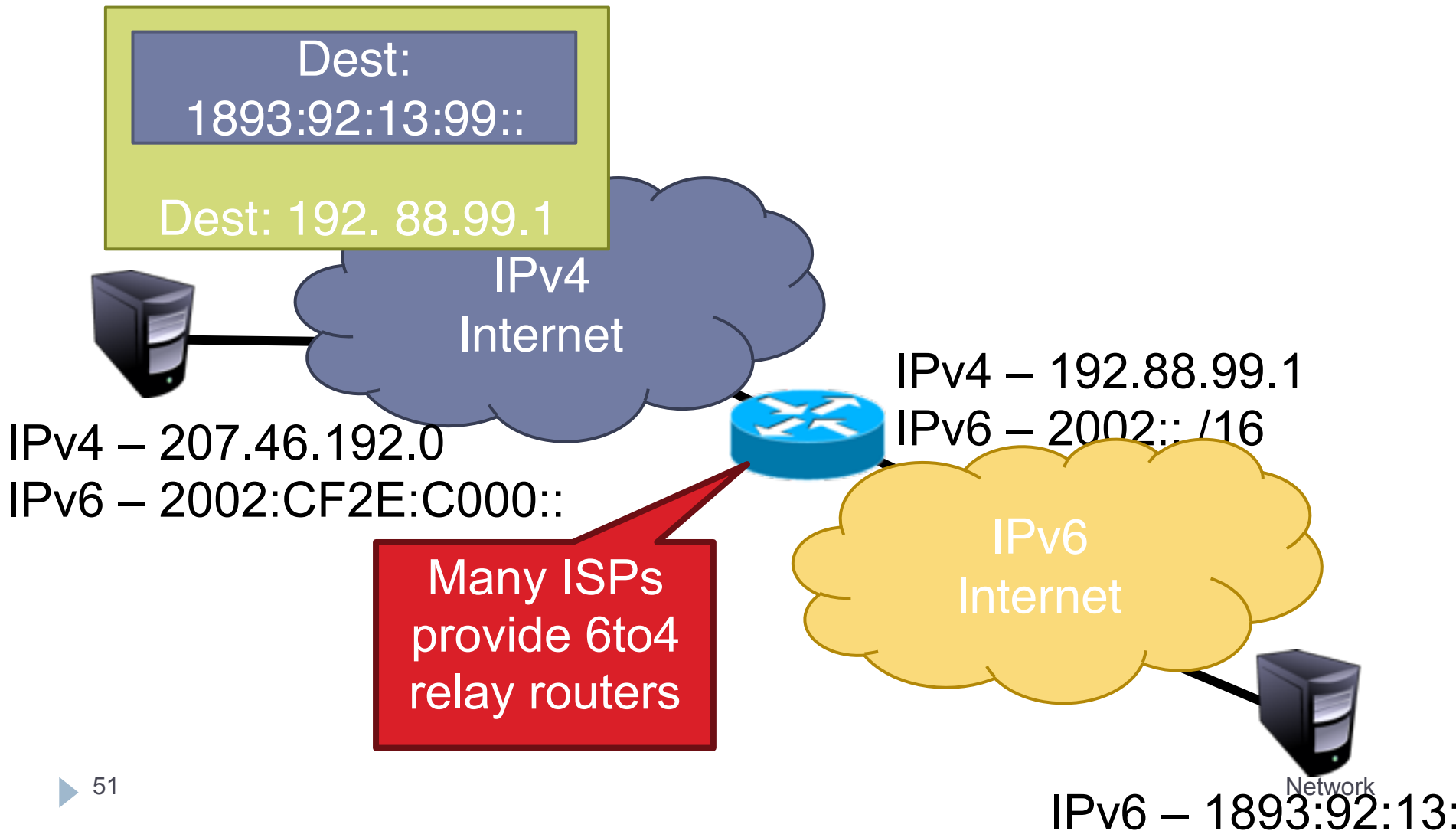
---



# Routing from 6to4 to Native IPv6

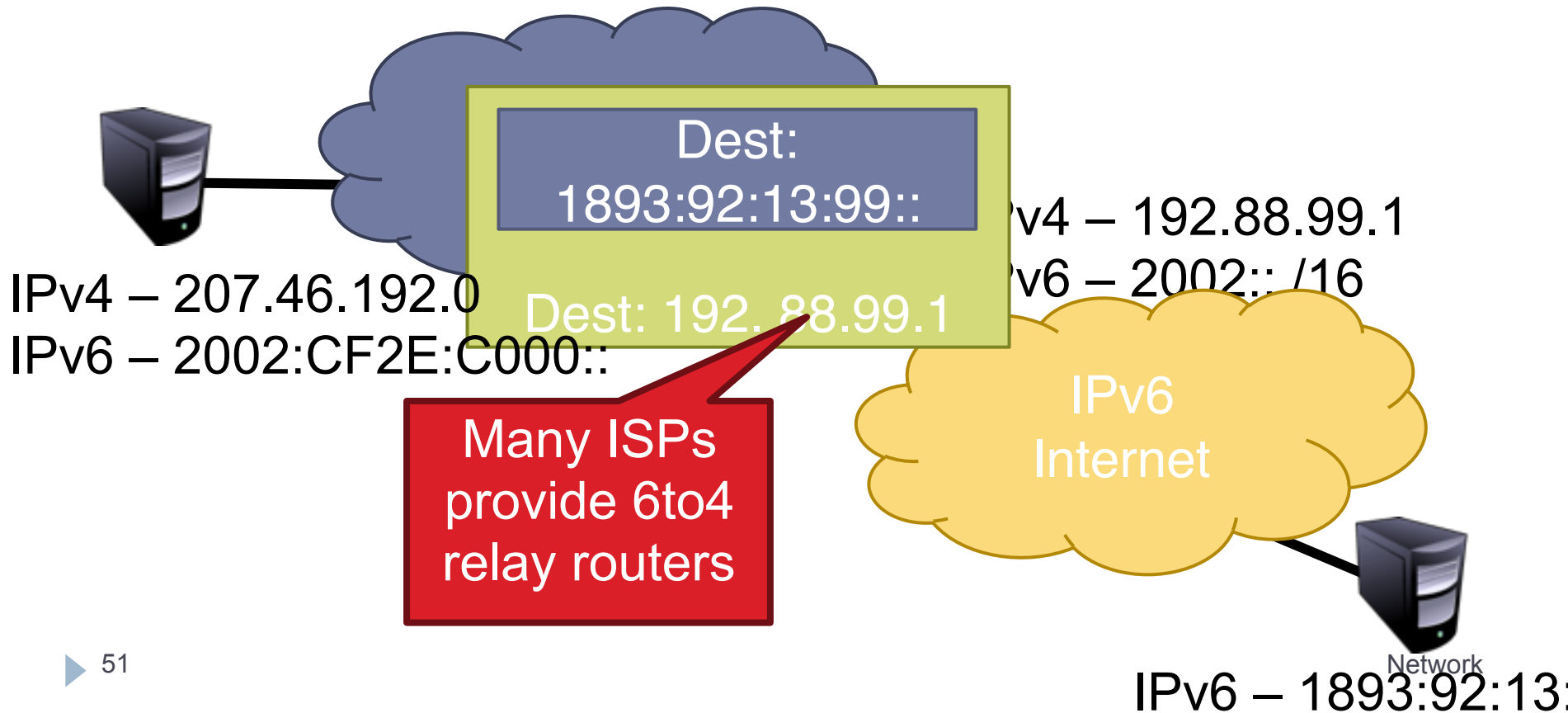


# Routing from 6to4 to Native IPv6

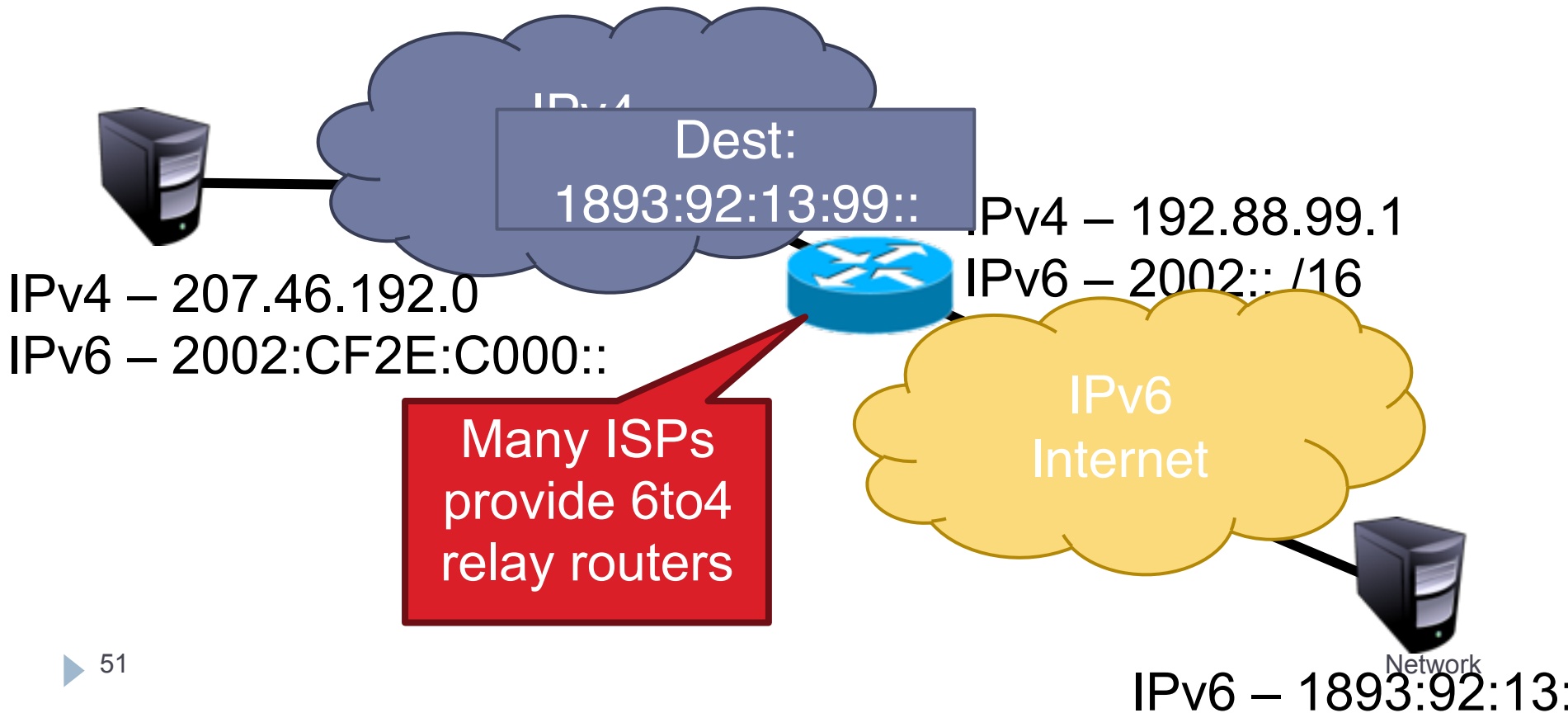




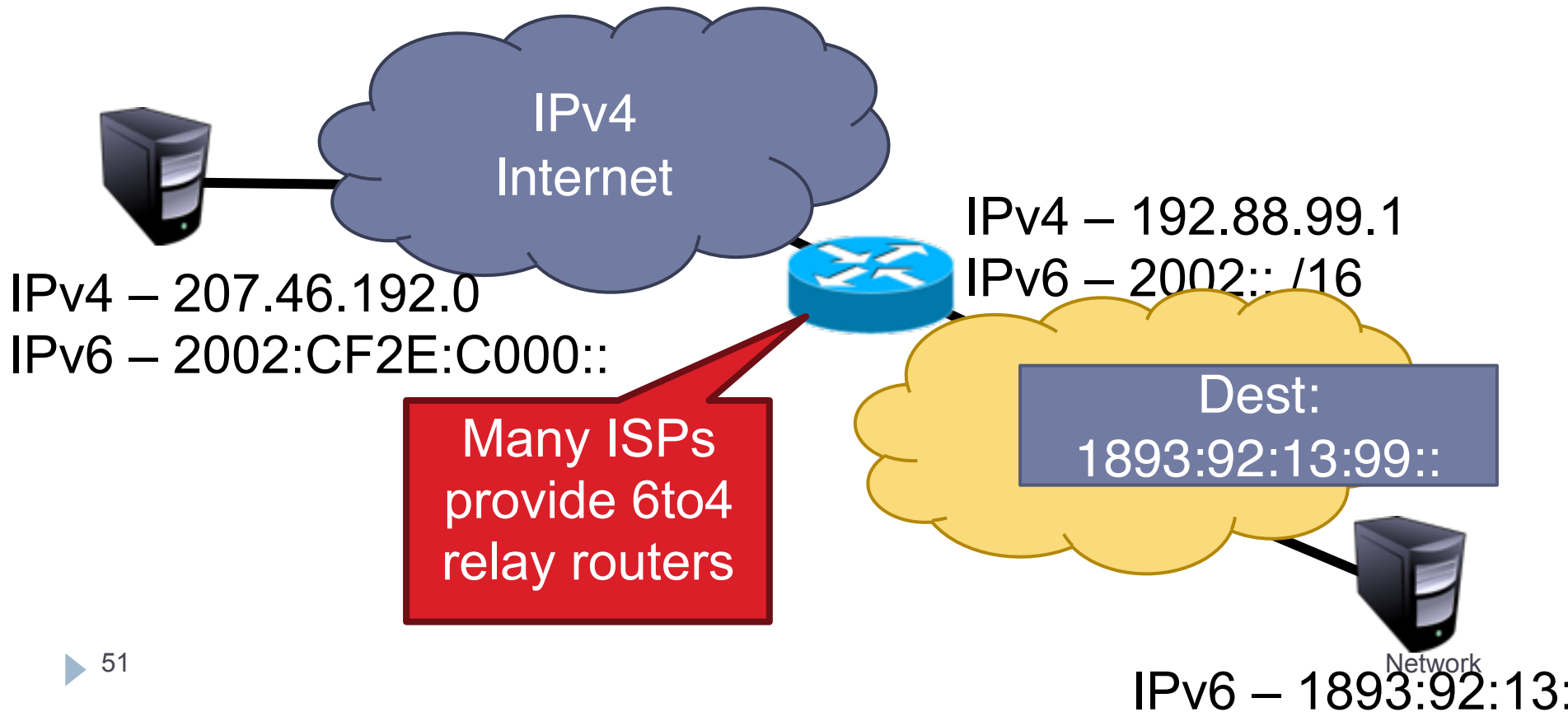
# Routing from 6to4 to Native IPv6



# Routing from 6to4 to Native IPv6

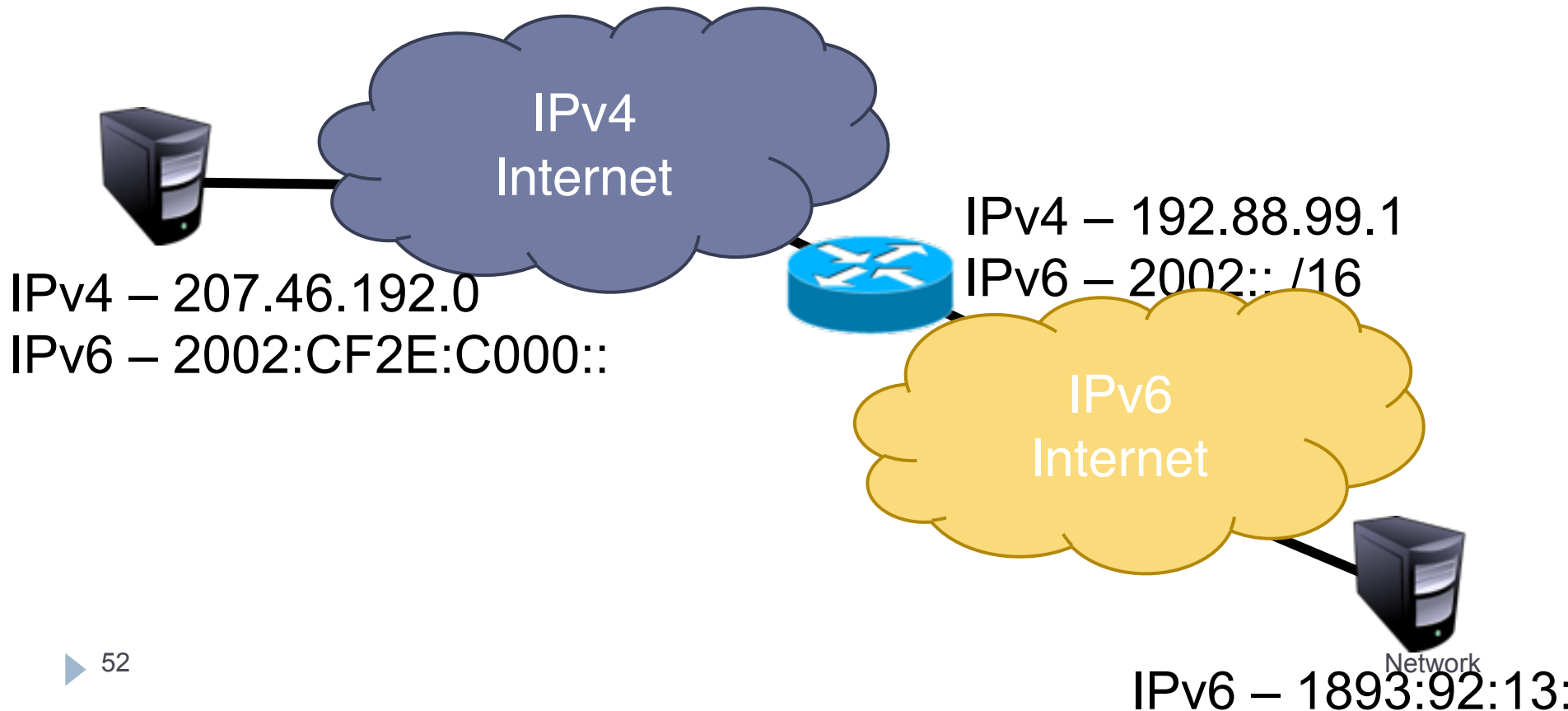


# Routing from 6to4 to Native IPv6



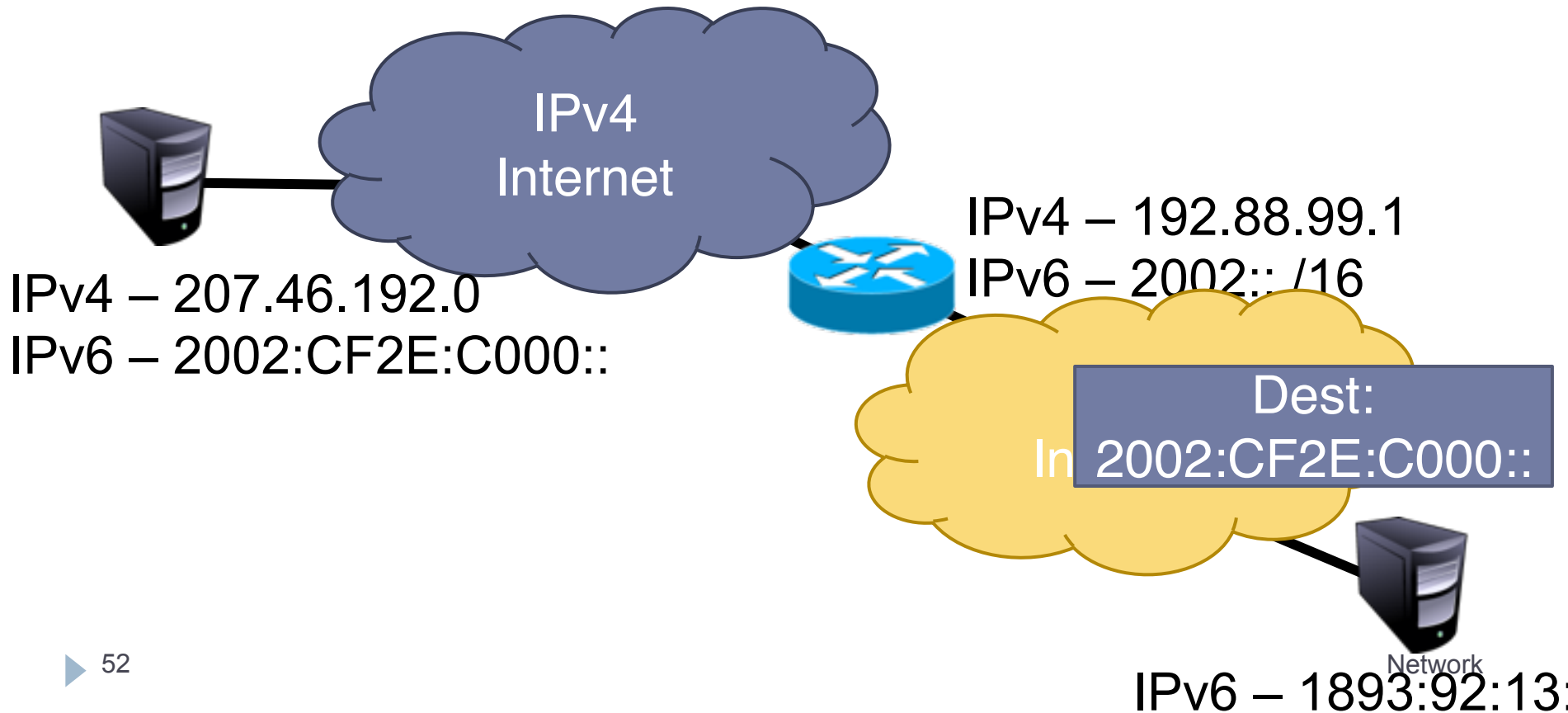
# Routing from Native IPv6 to 6to4

---

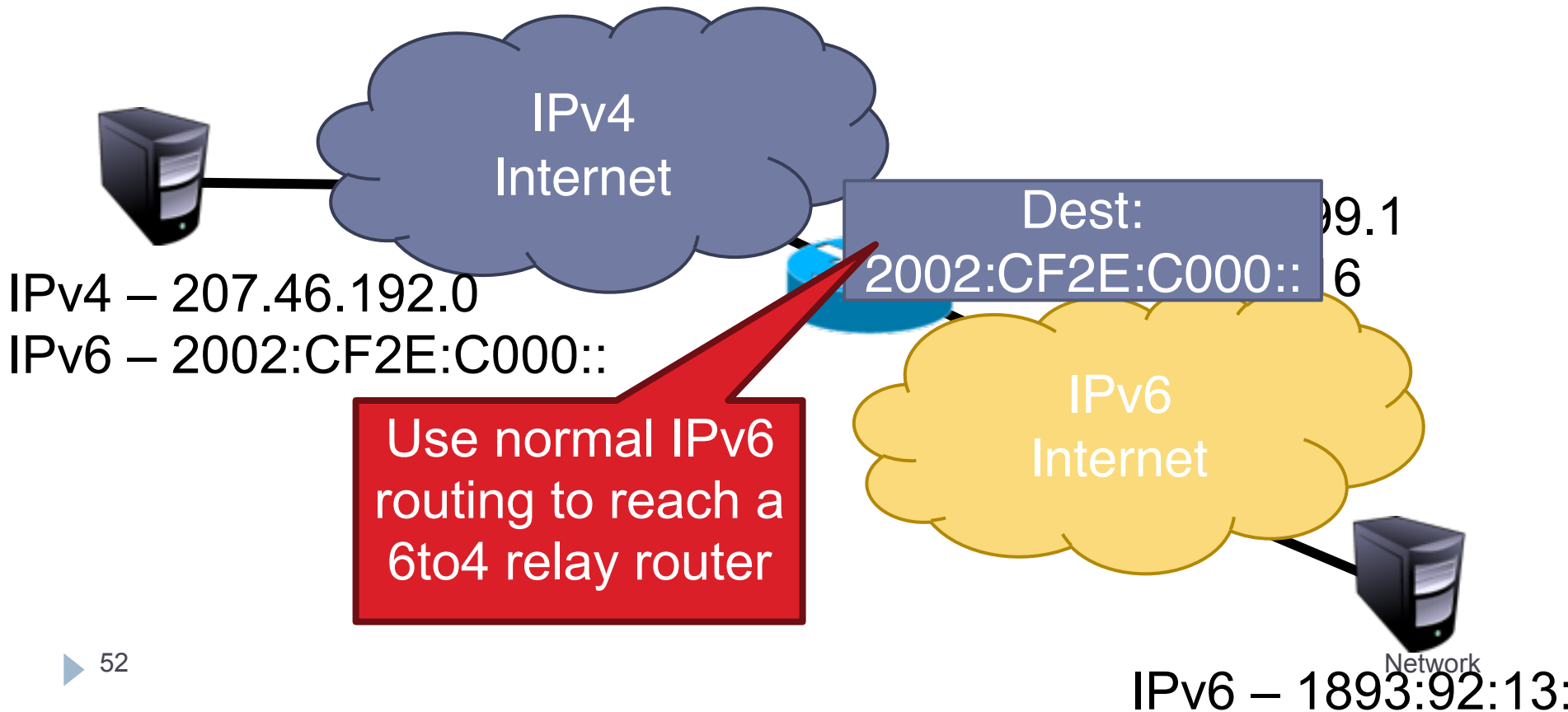


# Routing from Native IPv6 to 6to4

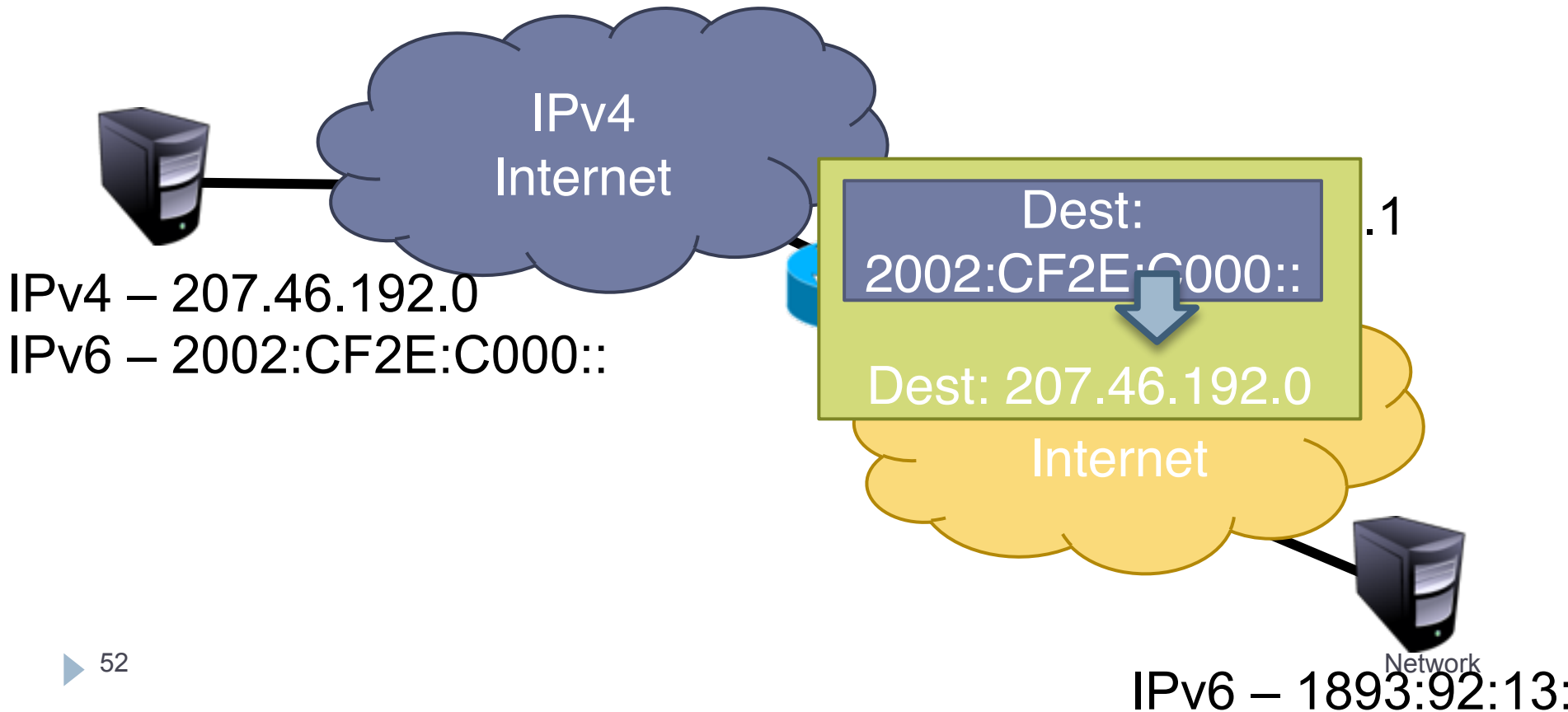
---



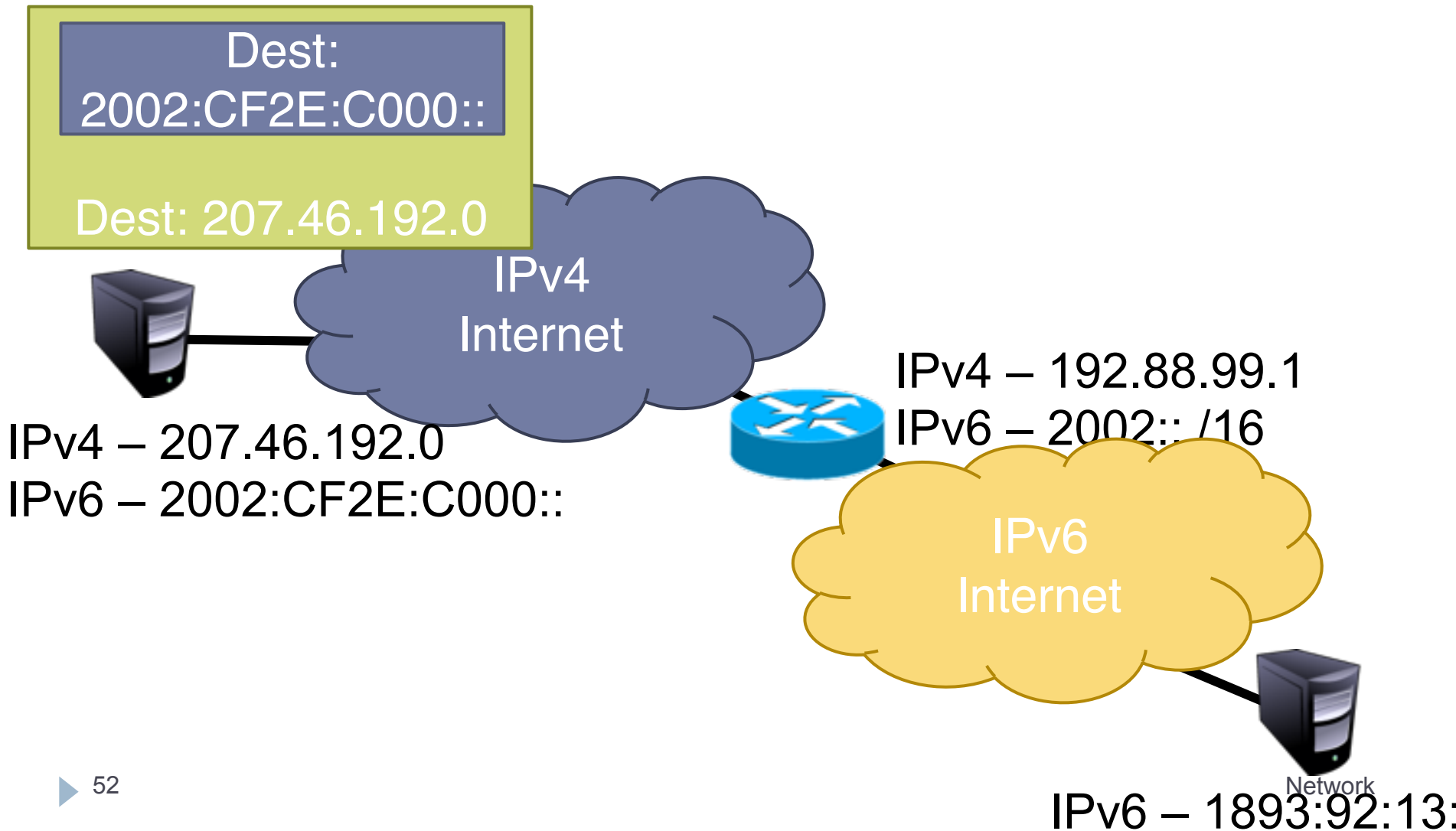
# Routing from Native IPv6 to 6to4



# Routing from Native IPv6 to 6to4



# Routing from Native IPv6 to 6to4

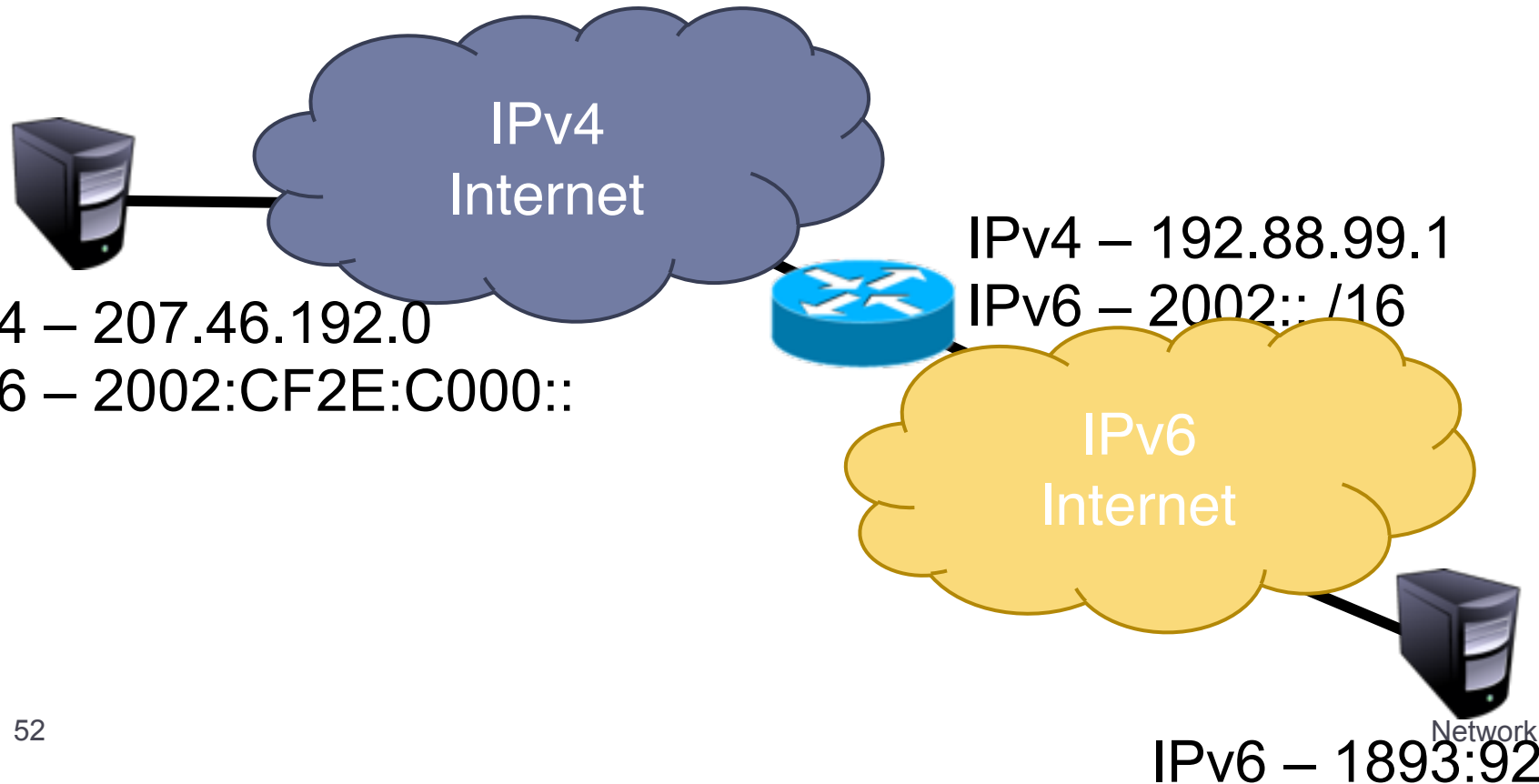




# Routing from Native IPv6 to 6to4

---

Dest:  
2002:CF2E:C000::



# Problems with 6to4

---

- ▶ **Uniformity**
  - ▶ Not all ISPs have deployed 6to4 relays
- ▶ **Quality of service**
  - ▶ Third-party 6to4 relays are available
  - ▶ ...but, they may be overloaded or unreliable
- ▶ **Reachability**
  - ▶ 6to4 doesn't work if you are behind a NAT
- ▶ **Possible solutions**
  - ▶ IPv6 Rapid Deployment (6rd)
    - ▶ Each ISP sets up relays for its customers
    - ▶ Does not leverage the 2002:: address space
  - ▶ Teredo
    - ▶ Tunnels IPv6 packets through UDP/IPv4 tunnels
    - ▶ Can tunnel through NATs, but requires special relays

# Consequences of IPv6

---

- ▶ Beware unintended consequences of IPv6
- ▶ **Example: IP blacklists**
  - ▶ Currently, blacklists track IPs of spammers/bots
  - ▶ Few IPv4 addresses mean list sizes are reasonable
  - ▶ Hard for spammers/bots to acquire new IPs
- ▶ **Blacklists will not work with IPv6**
  - ▶ Address space is enormous
  - ▶ Acquiring new IP addresses is trivial